IEEE MULTIMEDIA 1070-986X/97/$10.00 © 1997 IEEE
Vol. 4, No. 4: OCTOBER-DECEMBER 1997, pp. 74-82

Editor: Peiya Liu
Siemens Corporate Research

Contact Braun at the IBM Networking Center, Vangerowstr. 18, D-69115 Heidelberg, Germany, e-mail t.braun@ieee.org.

Contact Standards editor Peiya Liu at Siemens Corporate Research, 755 College Road East, Princeton, NJ 08540, e-mail pliu@scr.siemens.com.

Internet Protocols for Multimedia Communications, Part II: Resource Reservation, Transport, and Application Protocols

Torsten Braun *IBM European Networking Center*

The first part of this article (*IEEE MultiMedia*, July-Oct. 1997) surveyed the evolution of Internet protocols and applications and described Internet Protocol IPv6 in detail. This part discusses new developments at upper layers that support real-time Internet multimedia such as audio and video conferencing and shared whiteboard applications.

Application-level framing (ALF), proposed in 1990 for protocol and application design,[1] now forms the basis for many new Internet protocols and applications, including Real-Time Transport Protocol (RTP) and Mbone multimedia applications. RTP supports real-time applications that adapt to changing network situations to maintain quality of service (QoS). The Resource Reservation Protocol (RSVP) provides new Internet services with higher quality than best-effort by means of resource reservations.

Application-level framing

The Transmission Control Protocol (TCP) does not support transfer of real-time data such as audio and video. Because TCP recovers packet losses by retransmissions to provide reliable sequenced service, real-time data delivery must wait for receipt of all retransmissions. This introduces intolerable delay in the (frequent) case of packet loss. User Datagram Protocol (UDP) avoids this drawback with simple datagram-oriented (but unreliable) service; applications can be put on top of UDP with additional functions integrated into the applications. The ALF concept also proposes integrating system functionality into communication applications to implement efficient systems.

However, integrating traditional transport protocol functions such as reliability support requires that applications control the network packet size. ALF proposes a single packet size for all communication stack functions, which increases system performance and allows more advanced, efficient implementation. To support interoperability among applications, ALF protocol frameworks are defined by specifying mainly the protocol data unit formats. ALF protocol frameworks do not define algorithms for flow control or retransmission strategies; these are based on application requirements.

Real-time applications such as audio or video conferencing tools are often based on RTP to work in loaded networks as long as a certain amount of bandwidth is available. These applications benefit from RTP's support of intra- and interstream synchronization and media detection. Multicast applications require reliable communication in UDP- or IP-based multicast scenarios. The ALF-based Scalable Reliable Multicast (SRM) framework proposes algorithms that increase reliability over unreliable UDP/IP communication systems. Though not yet standardized, RTP data format enhancements for SRM could include the required information elements for retransmission requests and retransmissions.

Also based on ALF, the Lightweight Session (LWS) model generalizes IP multicast to support collaborative applications such as audiovisual (A/V) conferencing and shared whiteboards. The LWS building blocks include IP multicast, timing recovery via receiver adaptation, and thin transport layers according to the ALF concept;[2] these add scalability, fault tolerance, and robustness to IP multicast. SRM basically extends the LWS concept to support reliable multicast data transfer.[3]

Real-Time Transport Protocol

RTP, developed by the Internet Engineering Task Force (IETF) Audio/Video Transport working group, provides a good example of ALF-based protocol design.

RTP packet formats

RTP provides basic packet format definitions to support real-time communication but does not define control mechanisms or algorithms. RTP is often integrated into the application's processing operations rather than implemented as a separate layer. The packet formats provide information required for audio and video data transfer, such as incoming video data packet sequence. Sequence numbers permit packet loss detection or determining the position of a video frame within an image.

Audio and video applications also require synchronization. For continuous media such as pulse code modulation (PCM) audio, data streams can be synchronized by sequence numbers, but noncontinuous data such as MPEG-compressed video require time stamps for resynchronization. The RTP header's time-stamp value may also help to synchronize different real-time data, for example, audio and video streams to achieve lip synchronization. The payload type field supports different encoding schemes by indicating what data type the RTP payload contains (for example, MPEG, H.261, or Motion-JPEG video). Figure 1 shows the RTP data format.

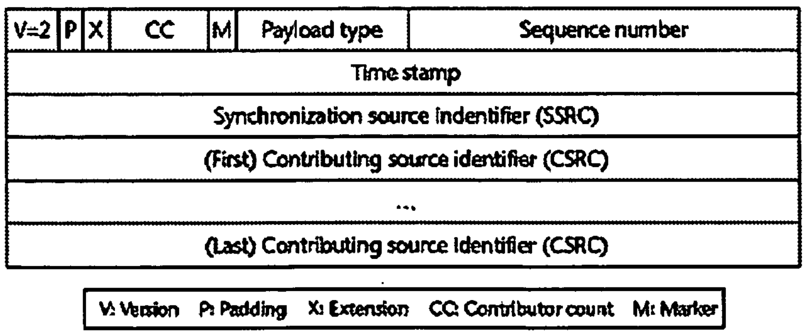
V=2
P
X
CC
M
Payload type
Sequence number
Time stamp
Synchronization source indentifier (SSRC)
(First) Contributing source identifier (CSRC)
...
(Last) Contributing source identifier (CSRC)
V: Version P: Padding X: Extension CC: Contributor count M: Marker

Figure 1. *RTP header format.*

Following the ALF principle, the semantics of some RTP header fields are application-dependent. Several profile documents specify header field use for different applications. For example, the marker bit defines the start of a talk spurt in an audio packet and the end of a video frame in a video packet.

Mixers and translators

Mixers and translators - application-level gateways operating on top of RTP - insert, delete, or modify an RTP packet's encoding. Modifying a flow's encoding scheme and changing the number of packets requires adapting the sequence numbers and possibly the payload type. While translators process different data streams independently, mixers may generate a single new data stream out of several different incoming data streams.

Each data stream originates from a source identified by the RTP packet's synchronization source (SSRC) identifier. Since a translator does not change the data stream's synchronization, it also does not change the SSRC identifier. A mixer combines several independent data streams to a single one, generating a new synchronized stream. The mixer therefore creates a new SSRC value and puts the old SSRC values of the combined contributing source data streams into the contributing source (CSRC) list.

For example, a mixer might combine several audio streams from an audio conference into a single stream. A mixer could also combine a hierarchically encoded audio or video stream consisting of multiple levels, each transferred in a separate stream. The mixer selects a subset of the levels and combines them into a single stream, enabling the limited-capacity receiver behind the mixer to receive the resynchronized stream.

Translators include audio/video converters, firewalls, and gateways that receive multicast audio data and forward data using unicast to nonmulticast receivers.

Real-Time Control Protocol

Like other protocols, RTP has a control protocol, real-time control protocol (RTCP). RTCP does not transfer data, just control information related to an RTP data stream. The application may define application-specific information elements.

The most important predefined control information elements are sender and receiver reports. *Sender reports* are generated by end systems that also send RTP data, *receiver reports* by receiving-only end systems. Both contain information about the data stream's current QoS plus *receiver blocks*, one for each data stream received. Each receiver block contains receiver statistics about the data stream such as highest sequence number received, interarrival jitter, fraction of lost packets since the last report, and cumulative number of lost packets. The sender report also includes sender information such as Network Time Protocol (NTP) and RTP time stamps and the number of packets and bytes already transmitted. The receivers use this information to calculate receiver block statistics.

RTCP packets also include *source descriptions* that describe the source in more detail. These can contain real user names, e-mail addresses, telephone numbers, geographic information, application names, and private extensions. This lets conference application programs using RTP/RTCP display information about participating conference users. Finally, RTCP uses *bye* messages to indicate that a user left the conference.

RTCP messages are transmitted to the same (unicast or multicast) IP address as the corresponding RTP data stream but with another port number (usually the RTP port incremented by 1). Adaptive applications frequently use sender and receiver reports to adapt their transmissions to current network conditions. When packet loss rates increase, possibly indicating an overloaded network, the senders can decrease their transmission rates. For example, video applications can switch to a more efficient, less bandwidth-consuming encoding scheme (Figure 2). When the network load declines (indicated by low loss rates), the senders can switch back to more bandwidth-consuming encoding schemes, which lower CPU overhead for compression and produce higher quality video.

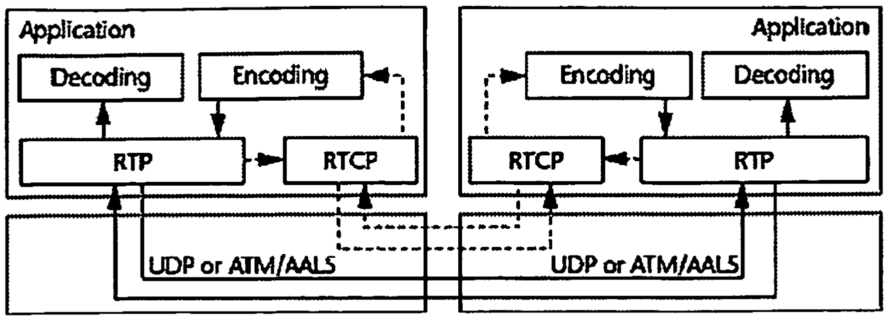

Figure 2. *Data transfer using RTP and RTCP.*

RTP usually runs over UDP/IP but can run over other protocols such as TCP/IP or even native AAL5/ATM. Running RTP over AAL5/ATM enhances the AAL5/ATM service for audio and video transfer. This makes sense when using an unspecified bit-rate service (UBR) or an available bit-rate service (ABR).

RSVP: A basis for the Internet Integrated Services Architecture

The Internet currently provides a pure best-effort service without any QoS guarantees. Traditional Internet applications such as file transfer, news, and electronic mail tolerate delays or recover packet losses. Application-level congestion control mechanisms or TCP's slow-start mechanism can reduce network congestion. Reducing the transmission rate as described above results in lower throughput and higher delay, which many new applications, especially multimedia, cannot tolerate. Interactive audio communication only works with an end-to-end delay lower than 200 to 300 milliseconds. Even the most efficient audio compression techniques cannot tolerate throughput values of much less than 10 Kbps to avoid packet loss. When packet loss or delay exceeds 10 to 20 percent, most users find the audio quality unacceptable.

Internet integrated services

Certain kinds of services require QoS guarantees, feasible with reservation of resources such as processing power and memory in end systems and intermediate systems such as switches, bridges, or routers. Network bandwidth must be reserved for applications with high throughput requirements. Before reserving resources, resource managers must ensure that enough resources exist to satisfy the resource reservation request (admission control). The resources must be assigned to the target application's data packets. The end system and router resource managers must also check that the data sent does not exceed reserved resources. Data exceeding the negotiated traffic specification can be discarded or processed with lower priority (policing) than data conforming to the traffic specification. Moreover, the data of different flows must be scheduled to provide each flow the resources required to achieve the desired service.

Resources are reserved according to a traffic specification based on a token bucket model. The token bucket consists of a token rate r that describes the token generation rate. The tokens are put at rate r into a token bucket of size b. Each byte sent consumes a token in the bucket (Figure 3). The peak bit rate p must not be exceeded. Any packet needs at least m (minimum policed unit) tokens to be transmitted even if the packet size is less than m bytes. The maximum packet size is limited by M (maximum packet size). During a time period T no more than $r * T + b$ data bytes may be transmitted. If the peak bit rate is defined, the amount of data that may be transmitted during T is also limited by $M + p * T$.

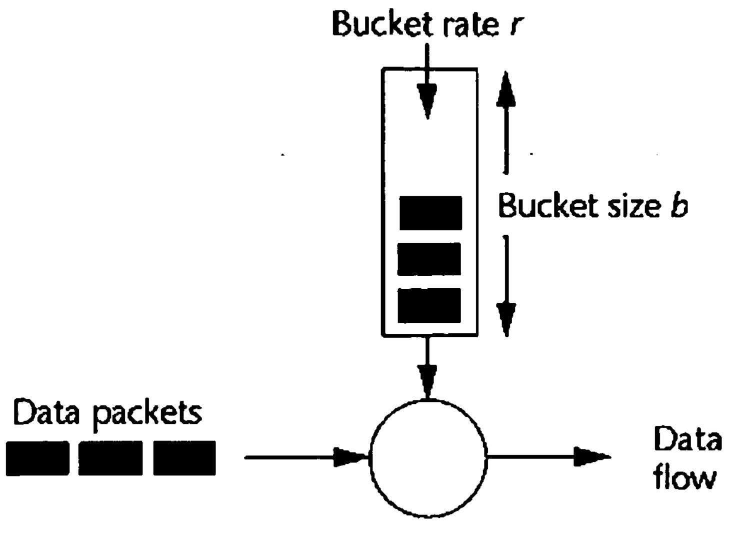

Figure 3. *The token bucket.*

In addition to traffic parameters *r*, *b*, *p*, *m*, and *M*, service-specific parameters such as desired type of service, bandwidth, and delay must be defined. The complete flow specification (FlowSpec) consists of the traffic specification (TSpec) and the service-specific requested service specification (RSpec).

The IETF Integrated Services working group has proposed several services, the most important being controlled-load service and guaranteed service. Applications using controlled-load service may assume successful delivery of most of the transmitted packets and that the packet delay does not significantly exceed the minimum delay between sender and receiver - service similar to that offered in an unloaded network. However, controlled-load service guarantees neither bandwidth nor delay; it is intended for Internet applications such as adaptive real-time audio and video that run well in slightly loaded networks but not in congested networks. To support controlled-load service, the end systems and routers perform admission control functions to avoid network overload and congestion.

Guaranteed service ensures both bandwidth and delay, and targets real-time applications with hard delay requirements such as audio and video playback. The requested service specification (RSpec) contains the requested service rate *R* and a slack term *S* that describes the allowed difference between desired delay and that resulting from *R*. Network elements delay a flow by *b/R*. They can use *S* to support a flow with a service rate lower than *R*. The delay *b/R* may be exceeeded by error terms *C* and *D*, where *C* depends on *R* and *D* is a constant value. The delay of a packet experienced in a network element is thus bound by *b/R* + *C/R* + *D*. The total end-to-end delay depends on the number of network elements, traffic specification parameters, and sums of *C* and *D* over all network elements between sender and receiver.

Implementation framework

A network element implementation consists of several components, depicted in Figure 4. The *packet classifier* filters packets belonging to certain flows and delivers them to the *packet scheduler*, which schedules the packets to correctly assign the reserved resources. For example, in a router, received packets must be stored in buffers and processed by the CPU. The scheduler must ensure sufficient buffers to store the packets and guarantee that the CPU processes the packets as needed to achieve the desired QoS. In a network environment such as Asynchronous Transfer Mode (ATM), where the network supports QoS, the scheduler also maps packet flows to ATM connections established with appropriate QoS parameters.

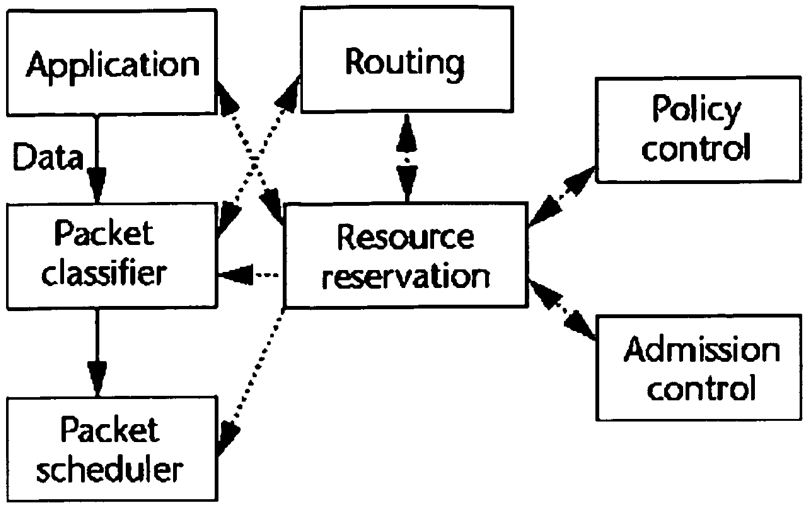

Figure 4. *Implementation model for a network element.*

The resource reservation entities exchange information using a protocol such as RSVP or the Stream Protocol Version 2 (ST-2). For an incoming reservation, the resource management system's admission control must ensure sufficient resources to support the specified QoS. The policy control must then check whether administrative conditions such as access control or accounting prohibit the reservation. If both tests are positive, the resources can be reserved and the flow's QoS requirements will be supported.

Resource Reservation Protocol

RSVP permits reserving resources for an IP packet flow without establishing an explicit connection.[4,5] RSVP also supports IP multicast communication and is receiver driven. As with the IP multicast model, where the receiver joins a multicast group, the RSVP receiver determines the resources to be reserved for a given flow.

RSVP supplements the IP protocol stack; it is used only to exchange signaling information between IP systems. This consists mainly of traffic descriptions that provide resource reservation information. IP end systems and routers use this information to reserve the resources required for the desired QoS. The RSVP systems (IP end systems or RSVP-capable routers) calculate required buffer space, CPU cycles, network bandwidth, and other resources from the exchanged signaling parameters.

RSVP uses path and reservation messages (PATH and RESV). The IP flow source transmits PATH messages to the data packets' IP destination, ensuring the PATH messages are transmitted on exactly the same path as the data. This approach also guarantees that non-RSVP routers forward PATH messages; if such a router does not recognize incoming PATH messages, it just forwards them along the normal routing path. An RSVP-capable IP router recognizes the PATH packet, stores the previous hop's IP address (PHOP) in an internal data structure called *path state*, and replaces the PHOP value in the PATH message with its own IP address. It then transmits the updated PATH message to the IP destination address. In addition to the PHOP value, the PATH message contains a sender template, a sender traffic specification, and an optional advertisement specification.

The *sender template* defines the data's sender and describes a subset of packets belonging to a session. A *session* is the set of packets with the same IP destination address and TCP/UDP port number. It may consist of several flows, each identified by a *filter specification*. The filter specification used in RESV messages has the same format as the sender template in PATH messages, consisting of the IP source address and the source TCP/UDP port number. IPv6 sender templates may optionally consist of the IP source address and the IPv6 flow label. Hence, an IPv4 flow is described by IP source address, source port, IP destination address, and destination port, while an IPv6 flow can be identified by IP source address and IPv6 flow label - that is, IP information only.

The *sender traffic specification* (Sender TSpec) describes data flow characteristics. The reservation depends on the flow specification in the RESV messages, but that may be chosen based on Sender TSpec or not. The receiver can analyze Sender TSpec and reserve exactly the resources receivable without any service degradation.

The *advertisement specification* (AdSpec) describes some characteristics and parameters of the sender-receiver path. Intermediate RSVP systems may change AdSpec parameters. For example, AdSpec may indicate whether all routers on the path support the different service types. Other AdSpec parameters include the number of RSVP routers on the path, estimated available bandwidth, minimum end-to-end delay, path maximum transmission unit (MTU) size, and sums of error terms *C* and *D* required for the guaranteed service.

Receiving end systems generate the RESV messages and send them back along the data's path - all RSVP systems know the previous hop of the PATH messages and data. RESV messages, therefore, are transmitted between the RSVP systems and always contain the previous hop's IP address as destination address. RESV messages also contain the flow specification (TSpec and RSpec) to be reserved. With a multicast data flow, several receivers might generate different flow specifications. A router receiving different specifications from receivers or down-stream routers for a single flow must merge them to a single flow specification, which is inserted in the RESV message sent to the previous hop (Figure 5). The maximum values of several flow specifications are used for the merged flow specification.

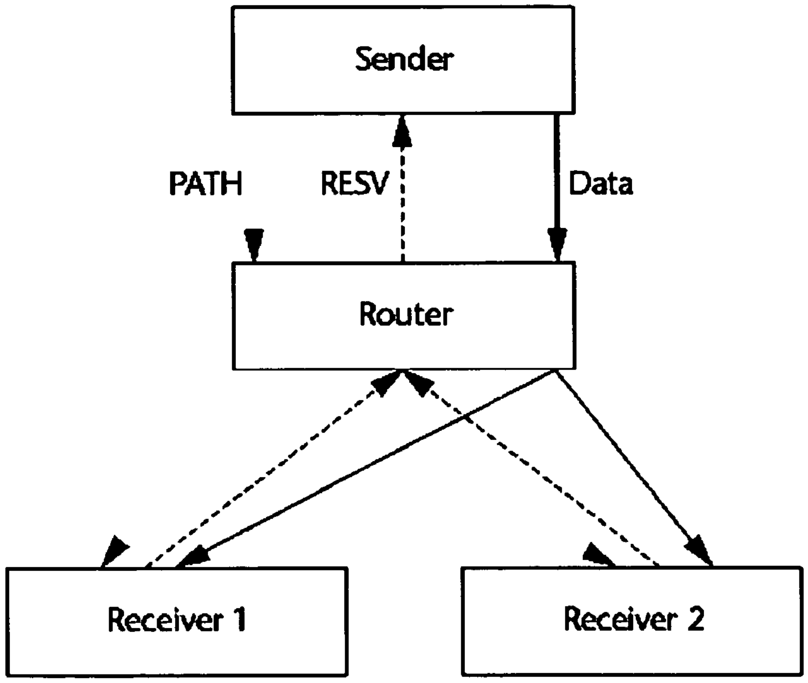

Figure 5. *RSVP message exchange.*

RESV messages also contain a filter specification describing the packet subset of a session for which the resources are reserved. It has the same format as the PATH messages' sender template, but a receiver can choose among several reservation styles. The *fixed filter* (FF) style reserves resources for exactly one sender and supports sessions with several simultaneously active senders, for example, a video conference in which all participants can see each other. The *shared explicit* (SE) filter style and *wildcard filters* (WF) are used to reserve shared resources for a set of senders. The SE style reserves resources for a known set of senders; the filter specification must specify each sender. The WF style reserves resources for arbitrary sender sets. SE and WF styles work when only one sender is active at a time, such as in an audio conference.

Periodic PATH and RESV message exchange increases reliability in several ways:

- Lost RESV messages are recovered by repeating them periodically.
- If a receiver changes the reservation parameters, the next RESV message will contain the updated parameters.
- If the path from sender to receiver changes, resources can be reserved along the new path; these are released if the system does not receive RESV messages within a certain time period.

The reservation states therefore adapt automatically to changing parameters; it is not necessary to establish and release resource reservations explicitly. This so-called soft state approach contrasts with the hard state approach used to establish connections in telephone or ATM networks.

Internet multimedia applications

The World Wide Web and real-time data transfer are the most popular emerging Internet applications. However, because they are based on different protocol architectures, it is difficult to integrate audio and video distribution over the Web.

Application protocols for the Web

The Hypertext Transport Protocol (HTTP) 1.0 is an application-level protocol that runs over TCP or any reliable transport protocol. HTTP uses a client-server mechanism: A client such as a WWW browser asks a WWW server for information using the GET request method; HEAD retrieves meta-information such as modification dates. PUT transfers information from client to server.

Each request creates its own TCP connection. When the client and server have exchanged the request and response messages, the TCP connection closes. A new request requires another TCP connection; HTTP thus establishes a new TCP connection for each WWW document and graphic. The clients and servers need not administer any state information since HTTP is a completely stateless protocol. Some applications such as Internet shopping require state information so that a server can map different requests to a single client. This requires transfer of state information such as a client identifier with each request and response.

This approach is obviously inefficient, since each new TCP connection costs one round trip in time.[6] Some browsers reduce this drawback by establishing several parallel TCP connections, but this may result in many open connections at a server. The TCP options for transactions (T/TCP) could avoid these problems. HTTP 1.1 now permits a single TCP connection for several HTTP transfers by defining additional methods such as OPTIONS, PUT, DELETE, and TRACE.

To transfer A/V data over the Web, a simple approach uses HTTP to download a file from a Web server; the client then starts a helper program to play the sound or video locally. This introduces very large delays, since large A/V files may take several minutes to download. Starting data output as it arrives at the client can reduce this delay. This involves exchanging only control data over HTTP/TCP between Web client and Web server, providing initial information about the real-time data to be transferred. The Web browser then launches the helper program (usually a browser plug-in) according to the parameters retrieved from the Web server via HTTP. These parameters may include content information, the data's encoding type (such as MPEG video or PCM audio), or the A/V server address to contact for retrieval.

The A/V helper program and server run a protocol to exchange control information required to start and stop data transmission. For example, the Real-Time Streaming Protocol (RTSP) supports creating VCR-like commands such as play, forward, rewind, pause, stop, and record. RTSP can control one or several time-synchronized continuous media streams and may act as a network remote control for multimedia servers. It runs over TCP or UDP and is not connection-oriented; RTSP sessions are labeled by session identifiers.

The A/V data are received by an A/V client program that may be identical with the helper program or run on a separate machine. The client program can then output the data as it arrives at the client. The data itself is transferred using a protocol other than RTSP or HTTP, for example, RTP/UDP. The control protocol (RTSP in this case) can negotiate the transfer protocol with the A/V server.

Using two different protocols to communicate with the A/V server permits redirecting the server's output to another destination address, different from the client address. A client could retrieve A/V information via HTTP from a Web server, such as an A/V file's type and contents. The client's A/V helper program then instructs the server to send the data to a multicast address or to the unicast address of the machine with the A/V client program (Figure 6). For example, a teacher could ask a Web server for interesting learning videos on a certain subject; the server returns several appropriate video server addresses with the names and identifiers of A/V files. The teacher's helper program then contacts an A/V server via RTSP and instructs it to transmit the A/V file via RTP/UDP to the multicast group address specifying the set of learners. RTSP can also instruct an A/V server to record A/V sessions such as a lecture, initializing the server and instructing it to record data transmitted from an A/V client to a multicast address.

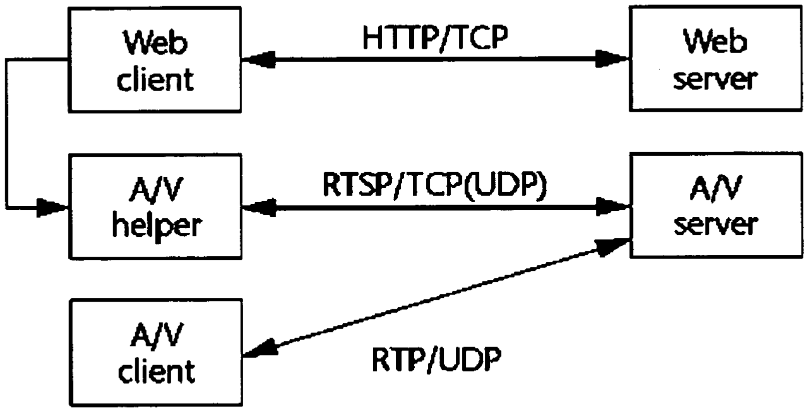

Figure 6. *Real-time data transfer over the World Wide Web.*

Mbone multimedia conferencing applications

Many multimedia applications have been developed for the Internet, the Mbone tools being among the most popular. A transmission of an IETF meeting in 1992 was the first Internet transmission of a big event, and in 1994 a Rolling Stones concert was multicast over the Internet Mbone. Live videos from NASA shuttle flights and even surgeries have been multicast. Today, these tools most often support transmissions of university lectures and teleseminars.

Xerox PARC's network video (NV) tool[7] and the Inria video conferencing system (IVS)[8] were the earliest video tools (IVS also supports audio transfer). The video conferencing tool Vic[9] has been implemented at Lawrence Berkeley Laboratory (LBL) based on experiences with NV and IVS. Today Vic is the most widely used video conferencing tool over the Mbone.

The Network Voice Terminal (Nevot)[10] from University of Massachusetts and the visual audio tool Vat[11] from LBL are probably the most well-known audio tools. Most are based on RTP and use a standardized packet format to encapsulate different audio and video encodings and transmit them over UDP/IP. These tools significantly influenced the RTP standardization process such that most can now interoperate.

Several applications use RTCP to adapt the transmission parameters from RTCP's QoS status reports. For example, IVS decreases the transmission rate if the median loss rate over all receivers exceeds the tolerable value. It increases transmission if the median loss remains below the tolerable value.[8] Inria's FreePhone audio conferencing tool combines rate control and forward error correction. Dependent on the RTCP QoS status reports, the tool adapts the amount of redundant information in audio packets and the transmission rate.[12]

Many Mbone tools available as binaries and source code allow researchers to perform experiments and create extensions. RSVP has also been integrated into some experimental conferencing tools such as Vic. For example, a video stream sender can use `PATH` messages to indicate the required bandwidth for high-quality video transmission. The receiver can derive the `RESV` message's QoS parameters from parameters contained in the `PATH` message.[13]

Another important conference component, distributed shared whiteboards, lets users share text and graphics. Wb is a widely used tool that allows any conference participant to create pages and draw on them; this requires close to 100-percent reliability. The generation of retransmission requests and retransmissions by SRM algorithms are integrated into the application. New retransmission requests and retransmission messages would enhance RTP and RTCP.

Several new applications have been developed around the original Mbone tools. The Mbone Video Conference Recorder (Mbone VCR)[14] provides functions to record and play back Mbone live sessions with multiple multicast multimedia data streams from different applications. During recording, Mbone VCR synchronizes the multimedia data streams based on information provided by RTP. To play back data streams, Mbone VCR transmits the recorded data using the original timing and packet format. The receivers have only to run an application such as vic or vat to watch or listen to the recorded data.

One RTP translator, video gateway (VGW),[15] transforms a high-quality, high bandwidth-consuming Motion-JPEG video stream to a lower quality, low bandwidth-consuming H.261 video stream. This transformation makes sense for heterogeneous receivers, some connected over high-bandwidth links to the Internet (such as university researchers) and others over a narrow-band ISDN connection to the Internet provider (such as home users). VGW can be used to transform the Motion-JPEG video to H.261 for private users while university users receive the high-quality Motion-JPEG video.

Mbone's popularity lets users choose among several live transmissions at once. Because a conference depends on several applications with parameters (such as IP multicast addresses and UDP port numbers) users might not understand, session directory applications such as SDR[16] simplify the creation of conferences by advertising different parameters such as applications required, audio or video encoding used, IP multicast addresses, and UDP port numbers. A user need only double-click on a conference to automatically start all required applications with the correct parameters.

Summary and outlook

The protocols described here are already in use or are being deployed on the Internet and in private intranets. In particular, RTP forms the basis for most real-time applications developed for the Internet and for various research and commercial multimedia projects. RSVP and IPv6, as part of future TCP/UDP/IP communication systems in workstations and personal computers, will require adapting applications to make use of features such as priorities and flow-based resource reservations.

IPv6 prototype implementations are currently being developed and tested against each other on diverse computer platforms. Almost all important router and end system vendors have at least prototype IPv6 implementations, generally provided free of charge for experimentation. Most end system and router manufacturers have begun to integrate IPv6 in their latest operating system versions; standard Internet applications will soon support both IPv4 and IPv6. The 6Bone (IPv6 backbone) has been established for global Internet tests, and it connects IPv6 routers and end systems equipped with prototype implementations and first products. Tunnels are set up between IPv6 systems over pure IPv4 networks.

RSVP implementations for routers and end systems will also come to market in 1998, with experimental versions already available. However, taking full advantage of RSVP requires modifying applications and inserting new function calls for the RSVP API to initiate resource reservations. Using RSVP to implement guaranteed services makes sense only if all routers on the path between source and destination and all end systems are RSVP capable. The many network technologies being used on the Internet do not support reservation mechanisms very well. An incremental introduction of RSVP therefore cannot support guaranteed services, though RSVP can be easily introduced in networks with a limited number of routers between end systems, such as intranets.

Finally, using RSVP might only truly work with tariffing - if resources can be reserved without fee, any user could arbitrarily reserve resources. LAN technologies such as Token Ring and Ethernet do not provide admission control or policing functions, necessary to provide guaranteed services. The IETF working group "Integrated Services over Specific Link Layers" (ISSLL) seeks solutions for this problem, although it remains difficult or even impossible to provide QoS guarantees over shared Ethernet networks because an end system can overload the network.

However, a trend toward switching-based LANs means only a few end systems or even just one is connected to an Ethernet or a Token Ring port of a LAN switch. In this case, implementing admission control and policing functions on the LAN switch would significantly improve performance. An ISSLL working group is investigating how to map Integrated Services parameters to link-level parameters of networks that support QoS guarantees such as ATM.

References

1. D. Clark and D. Tenenhouse, "Architectural Considerations for a New Generation of Protocols," *Proc. ACM Sigcomm 90,* ACM Press, New York, 1990, pp. 200-208.
2. V. Jacobson, "Multimedia Conferencing on the Internet," *Proc. ACM Sigcomm 94 Tutorial,* ACM Press, New York, 1994.
3. S. Floyd, V. Jacobson, and S. McCanne, "A Reliable Multicast Framework for Lightweight Sessions and Application Level Framing," *Proc. ACM Sigcomm 95,* ACM Press, New York, 1995, pp. 342-356.
4. L. Zhang et al., "RSVP, A New Resource ReSerVation Protocol," *IEEE Network,* Sept. 1993.
5. S. Thomas, *IPng and the TCP/IP Protocols,* Wiley Computer Publishing, New York, 1996.
6. H. Schulzrinne, "World Wide Web, Whence, Whither, What Next ?" *IEEE Network,* Vol. 10, No. 2, Feb. 1996, pp. 10-17.
7. R. Frederick, "Experiences with Real-Time Software Video Compression," *6th Int'l Workshop on Packet Video,* Sept. 1994, pp. F1.1-F1.4.
8. T. Turletti and C. Huitema, "Videoconferencing on the Internet," *IEEE/ACM Trans. on Networking,* Vol. 4, No. 3, June 1996, pp. 340-351.
9. S. McCanne and V. Jacobson, "Vic: A Flexible Framework for Packet Video," *Proc. ACM Multimedia 95,* ACM Press, New York, 1995, pp. 511-522.
10. H. Schulzrinne, *Voice Communication Across the Internet, A Network Voice Terminal,* Research Report, Univ. of Massachusetts, Amherst, Mass., July 1992.

11. V. Jacobson and S. McCanne, *VAT - LBNL Audio Conferencing Tool,* http://www-nrg.ee.lbl.gov/vat/ .
12. J.-C. Bolot and A. Vega-Garcia, "Control Mechanisms for Packet Audio in the Internet," *Proc. Infocom 96,* April 1996, pp. 232-239.
13. T. Braun and H. Stüttgen, "Implementation of an Internet Video Conferencing Application over ATM," *IEEE ATM 97 Workshop,* May 1997.
14. W. Holfelder, "Mbone VCR - Video Conference Recording on the Mbone," *Proc. ACM Multimedia 95,* ACM Press, New York, Nov. 1995, pp. 237-238.
15. E. Amir, S. McCanne, and H. Zhang, "An Application-Level Video Gateway," *Proc. ACM Multimedia 95,* ACM Press, New York, Nov. 1995.
16. M. Handley, *The SDR Session Directory,* http://mice.ed.ac.uk/mice/archive/sdr.html .

Technical Resources

For technical details about the protocols discussed in this article, refer to the following documents. You can obtain an RFC (Request for Comment) at ftp://ds.internic.net/rfc/ or ftp://ietf.org/internet-drafts/ .

T. Berners-Lee, R. Fielding, and H. Frystyk, *Hypertext Transfer Protocol HTTP/1.0* , RFC 1945, May 1996.

R. Braden, D. Clark, and S. Shenker, *Integrated Services in the Internet Architecture, An Overview* , RFC 1633, June 1994.

R. Braden, *T/TCP: TCP Extensions for Transactions - Functional Specification* , RFC 1644, July 1994.

R. Braden et al., *Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification* , RFC 2205, Sept. 1997.

L. Delgrossi, L.Berger, *Internet Stream Protocol, Version 2 (ST2+)* , RFC 1819, Nov. 1995

R. Fielding et al., *Hypertext Transfer Protocol - HTTP/1.1* , RFC 2068, Jan. 1997.

R. Hinden, *IP Next Generation* , http://playground.sun.com/ipng .

IETF, *Integrated Services over Specific Link Layers (ISSLL) Charter* , http://www.ietf.org/html.charters/issll-charter.html .

IETF, *Integrated Service (intserv) Charter* , http://www.ietf.org/html.charters/intserv-charter.html .

IETF, *Audio/Video Transport (AVT) Charter* , http://www.ietf.org/html.charters/avt-charter.html .

H. Schulzrinne, *RTP Profile for Audio and Video Conferences with Minimal Control* , RFC 1890, January 1996.

H. Schulzrinne et al., *RTP, A Transport Protocol for Real-Time Applications* , RFC 1889, Nov. 1995.

H. Schulzrinne, A. Rao, and R. Lanphier, *Real-Time Streaming Protocol (RTSP)* , Internet Draft, draft-ietf-mmusic-rtsp-05.ps, work in progress, Oct. 1997.

S. Shenker, C. Partridge, and R. Guerin, *Specification of Guaranteed Quality of Service* , RFC 2212, Sept. 1997.

J. Wroclawski, *The Use of RSVP with IETF Integrated Services* , RFC 2210, Sept. 1997.

J. Wroclawski, *Specification of the Controlled-Load Network Element Service* , RFC 2211, Sept. 1997.

Distribution of Digital Media in the Internet

Marko Mäkelä <msmakela@cc.hut.fi>

Lam Ngoc Minh <nlam@cc.hut.fi>*)

Helsinki University of Technology

Telecommunications Software and Multimedia Laboratory

Abstract

Since the World Wide Web boom in 1994-1995, Internet has become a distribution channel of digital media in the eyes of the large audience. Teleconferencing and other applications involving distribution of audio and video over the Internet require a powerful network infrastructure and sophisticated protocols. This paper sheds some light on ATM and multicasting, but the main emphasis is on protocols that control the distribution of real-time data streams. There is also some discussion of interactive non-real time applications and protocols.

*) Sections 1 through 3 and most of Section 8 were written by Marko Mäkelä. Sections 4 through 6 were written by Lam Ngoc Minh.

Table of Contents

1. Introduction

Internet has always been a distribution medium for information. In the pre-WWW days, as the Internet was not widely known, researchers and other people distributed their works mainly through mailing lists, Usenet News and FTP sites.

The World Wide Web information retrieval system was developed at CERN in order to make it possible to publish articles in hypertext format. The Hypertext Markup Language (HTML) allows much more flexible presentation of documents than purely textual mediums, such as mailing lists and newsgroups.

For teleconferencing, Internet has traditionally offered two applications: `talk` for character-based one-to-one communications, and IRC (Internet Relay Chat) for line-based many-to-many communications. Both applications are very widely used, but the big masses demand easier-to-use communications methods, such as Internet telephony or videoconferencing.

This representation concentrates on new digital media technologies and applications on the Internet. Most development was started after the big WWW boom, but there are some applications that have been under development for a longer time not only in the Internet, such as teleconferencing.

2. Real-Time Applications and Protocols

The trend for real-time applications, such as distribution of real-time video and audio streams on the Internet poses great demand on the network infrastructure and protocols. The TCP/IP protocol suite does not have any quality of service (QoS) guarantees, and the Internet was not originally built for transferring real-time media.

Lossy compression techniques have been developed in order to dramatically cut down the required data rate, and protocols have been developed to make bandwidth allocation and QoS guarantees possible.

2.1 IP over ATM

Real-time applications are very resource-hungry. In the area of networking, only ATM (Asynchronous Transfer Mode) technology seems to have enough capacity for high-end applications. It was not designed with Internet in mind, and thus interfacing the two together is not entirely trivial.

The current IP over ATM development revolves around cell switching routers, multicasting, and bandwidth allocation.

2.1.1 Cell Switching Routers

The importance of cell switching routers (CSR) [1] cannot be overestimated. Traditional routers have to queue incoming datagrams and to parse their headers in order to figure out where they should be sent to. This queuing and processing causes delay and jitter, which cannot be tolerated by real-time applications. Under heavy traffic, a router's message queue can become full, and further datagrams will be dropped. This is called congestion.

A cell switching router (CSR) is a combination of an IP router and an ATM switch. Between the CSRs, there are two kinds of ATM Virtual Connections (VCs). Default VCs are being used for occasional IP traffic, while Dedicated VCs will be allocated for specific IP flows, which can be identified by source or destination port numbers and addresses or by IPv6 flow labels. Data streams such as HTTP, FTP or SMTP can be assigned a bypass pipe through the ATM cloud, even across IP network boundaries. If there is traffic from the same source host or subnetwork to the same destination host or network, it can be routed through the dedicated VC.

Dedicated VCs reduce the processing overhead in the cell switching routers. They also make quality of service management much easier. A real-time stream can be given a constant bit rate (CBR) or variable bit rate (VBR), while an elastic stream, such as HTTP, copes with available or unspecified bit rate (ABR or UBR). For policy reasons, it is also possible to use default VCs for streams from certain hosts.

Cell switching routers can easily support IP multicasting on the ATM level. A source VC is simply connected to multiple destination VCs. This is interoperable with the current IP multicast over ATM technique. The CSR can also handle Resource Reservation Protocol (RSVP) messages and provide the required QoS.

2.1.2 Bandwidth Allocation

IP normally guarantees best-effort delivery, which is insufficient for real-time media. On the other hand, ATM has QoS classes for constant bit rate, real-time variable bit rate, available bit rate, and so on. [3] discusses service mappings, i.e. how the QoS parameters from e.g. RVSP should be mapped to ATM QoS classes, and how to balance real-time streams and elastic data bursts.

2.2 Multicasting: MBone

Class D addresses in the IP address space are reserved for multicasting applications. The addresses in this range (`224.0.0.0-239.255.255.255`) identify the multicasting groups. [4] is an excellent introduction to multicasting, to the Internet's Multicasting Backbone (MBone) and to multicasting routers.

MBone is probably best known for video broadcasts made with it, but it is merely a network service that can be used e.g. via UDP. The introductory document [4] uses distributing stock quotes as an example application. It also has a good definition of the MBone:

> The Internet Multicast Backbone (MBone) is an interconnected set of subnetworks and routers that support the delivery of IP multicast traffic. The goal of the MBone is to construct a semipermanent IP multicast testbed to enable the deployment of multicast applications without waiting for the ubiquitous deployment of multicast-capable routers in the Internet.

Not all Internet routers are in the MBone. Sometimes MBone must be tunneled to get it distributed over non-MBone routers. MBone routers use different multicast routing protocols. The main principle is to forward the packets away from their source until the TTL reaches zero.

2.2.1 IP Multicasting over ATM

Unlike Ethernet, ATM is not a broadcasting network. The hosts on an ATM network only listen to the VCs they are connected to. For this reason, special measures must be taken in order to make multicast and broadcast work in an ATM network.

The solution is Multicast Address Resolution Server (MARS) [2], a central server where every host that wants to belong e.g. to the broadcast group `255.255.255.255` in the logical IP subnet will register itself. The MARS keeps track of the multicast groups, and it can either reply with a list of the VPI/VCIs belonging to a group, or it can use a Multicast Server (MCS) which sets up a point-to-multipoint VC, and return the address of that VC. The use of a MCS poses less load on the multicasting host, and it is the desirable way of multicasting digital media applications.

2.2.2 Announcing Conferences: SAP and SDP

Radio and TV stations announce their broadcasting schedule in newspapers. That would not be feasible with MBone sessions due to the real-time and on-demand nature of the Internet. Instead, two protocols are used for announcing multicast sessions.

The Session Announcement Protocol (SAP) is used for announcing multicast conference sessions. The protocol is described in [5]. An SAP client announces the session on a well-known multicast address with the same scope (group address range or TTL parameter) as the conference session. This ensures that only hosts that will be able to receive the session get the announcement. SAP builds on the User Datagram Protocol.

SAP supports encryption and signing of the announcements, so it has some protection against spoofing and forgery. Unencrypted session announcements without a digital signature are quite much at the mercy of attackers. The digital signature (an one-way hash function) makes forgery practically impossible. If the multicast session is confidential, the announcement can be encrypted. The key identifier field tells the audience which keys to try. Yes, there may be several keys with the same key identifier, because the IDs won't be administrated globally. If more confidentiality is wanted, then also the multicast session in question should be encrypted. For this reason, the payload message can contain the session encryption key.

SAP announcements will be sent periodically. Any host that receives the announcement will add it to its session list, provided that it is not encrypted or it can be decrypted, and its authentication header (if any) is valid. Sessions will be deleted from the list either by sending a deletion request, or by letting the session time out.

The payload data in the SAP announcements are in the Session Description Protocol (SDP) [6] format. SDP messages are not limited to SAP distribution; as plain text messages, they can also be published via e-mail or on a Web page. SDP defines five compulsory data fields and lots of optional fields. The compulsory fields are: protocol version, owner/creator and session identifier, session name, time the session is active, and media name and transport address. Optional fields include contact addresses and all kinds of information. The fields have one-character identifiers, and the field names and values are separated with an equal sign (=). The format is very compact but still human-readable. Figure 2.1 illustrates this.

Figure 2.1: An SAP/SDP example

UDP header

SAP header

- Message type: 0=announce, 1=delete
- original source
- optional authentication header

SDP payload

```
v=0
o=msmakela 2908448562 2988024087 IN IP4 130.233.17.141
s=Internetworking Seminar
e=msmakela@cc.hut.fi
t=2873397496 2873404696
a=recvonly
m=audio 3456 RTP/AVP 0
m=video 2232 RTP/AVP 31
m=whiteboard 32416 UDP WB
...
```

2.3 The Real Time Protocol (RTP)

The Real Time Protocol (RTP) [9] is one of the few Internet standards concerning real time data transfer and discussing quality of service (QoS), the magic word of real time applications. Real Time Protocol itself does not make any service guarantees, but its accompanion protocol, the RTP Control Protocol (RTCP), monitors the quality of service and provides feedback to the application for congestion control.

A good description of RTP is in the abstract of the memo [9]:

> RTP provides end-to-end network transport functions suitable for applications transmitting real-time data, such as audio, video or simulation data, over multicast or unicast network services. RTP does not address resource reservation and does not guarantee quality-of-service for real-time services. The data transport is augmented by a control protocol (RTCP) to allow monitoring of the data delivery in a manner scalable to large multicast networks, and to provide minimal control and identification functionality. RTP and RTCP are designed to be independent of the underlying transport and network layers. The protocol supports the use of RTP-level translators and mixers.

RTP builds upon UDP, the connectionless transport protocol for both unicast (point-to-point) and multicast (point-to-multipoint) connections. An RTP network consists of endpoints, which are normal workstations, translators, which are a kind of RTP repeaters, and mixers, which mix data from several sources.

Mixers perform conversions between different media encoding formats and mix data from multiple sources together. As the Internet is very heterogenous in bandwidth, the participants of an RTP-based conference can be connected at very different speeds. Some participants sit on a high-speed local area network, while some have a slow modem or ISDN connection. It would not be desirable to limit the video and audio quality of all participants to the level tolerated by the lowest common denominator. RTP mixers convert stream formats on the fly and reduce the data flow from the faster network so that it will fit on the slow connection. Mixers, as their name hints, may also mix several audio and video streams together and multicast them to all participants.

Translators are required for tunneling RTP traffic through firewalls, or for making protocol conversions between IP-based and other networks. Some translators pass the payload data through untouched, others may change the encoding or e.g. add or remove encryption. In any case, translators are quite transparent for the end points, because the data will be sent with the original sender identifiers.

Mixers have one advantage over translators: they reduce the data flow. For instance, in an audio conference with several participants, several audio streams can be mixed into one, and each participant will receive only one audio stream from the mixer instead of one stream from each party.

The Real Time Protocol itself has been kept very simple in order to reduce processing overhead. The standards track defines a variable-length header extension field, which can be easily ignored by hosts that do not support extensions. The header always includes payload type, sequence number, timestamp and the synchronization source (SSRC) identifier, which identifies the sender. A mixer may add up to 15 contributing source (CSRC) identifiers to the header, thus indicating the original data sources. A marker bit can be used in an application-specific way, e.g. for marking frame boundaries in a video stream.

RTP sessions are controlled with the RTP Control Protocol (RTCP). For QoS monitoring, there are two messages that pass statistical data: sender report (SR) and receiver report (RR). Source descriptions (SDES) containing information about the data sources will be sent periodically, so that new receivers can join the conference on the fly. RTCP has extensibility through one application-specific message, APP. Last but not least, participants can leave the conference by sending a BYE packet.

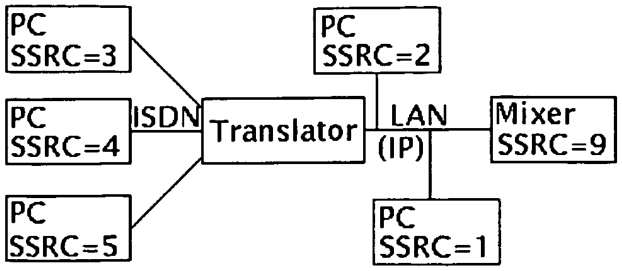

Figure 2.2: An RTP-based setup

2.3.1 HTTP over RTP

The Hypertext Transfer Protocol (HTTP) is the core protocol of the World Wide Web. It is designed for on-demand retrieval of documents, and as a TCP-based protocol, it is limited to unicast connections. However, there are applications that require real-time one-to-many communications, such as stock ticker updates sent to stock dealers all over the Internet.

The Internet draft [10] defines an HTTP payload format for RTP [9]. It is very straightforward; the HTTP stream is basically in the same format as in normal HTTP. Due to the unidirectional nature of multicasting, the server won't get any requests from the clients, and it must identify the documents it serves by sending a preamble header with a GET «request» specifying the Uniform Resource Identifier, or URI. The clients will cache the documents, which is useful in case of transmission errors. Remember, RTP is an unreliable protocol.

Several documents can be sent in a single RTP stream, with the same source identifier. The document (as identified by the URI) will be sent periodically, and it may change between retransmissions. The documents can be transmitted on several multicast groups, and the receiver can pick the group that is most suitable to its available bandwidth. Not only the data rate varies from group to group; some groups may provide more error correction information with redundant coding.

Receiver reporting can be used for automatical adjustment of transmission parameters or for gathering information about the listenership for business purposes. It can be turned off when it is not desirable. For instance, when the receiver is connected via a satellite link, the delay would be too long. Also sender reporting is possible: the sender may provide information about the resource being transmitted or about the status of the transmission.

The HTTP multicast session can be synchronized with other media, such as a video or audio stream from a presentation. Just like any multicast session, it can be announced with SAP and SDP (Section 2.2.2).

2.4 CSCW, Multimedia Teleconferencing

One of the most common computer supported collaborative work (CSCW) forms is multimedia teleconferencing. This involves transferring real time video and audio streams as well as updating e.g. a virtual whiteboard or a shared object desktop.

2.4.1 ITU-T T.120, Real Time Data Conferencing

The Telecommunication Standardization Sector of the International Telecommunication Union (ITU-T) has defined a set of protocols for real time data conferencing called T.120. The full specification is available from ITU for charge. This presentation is based on DataBeam Corporation's primer on the T.120 standards [7].

T.120 provides means for reliable network and platform independent multipoint data delivery. It is also network transparent: it does not matter if you have a dial-up modem connection or a fast LAN. T.120's support for varied topologies (Figure 2.4) makes it fairly straightforward to implement on the Internet by mapping the Multipoint Control Units (MCU) to routers. T.120 is scalable and extensible, so it can be adapted for new transport stacks easily. While it was developed for teleconferencing, it can be also used for other purposes, such as interactive games or virtual reality. T.120 cross-references some other ITU-T standards such as the H.32x series for video conferencing.

The T.120 protocol stack (Figure 2.3) consists of three low-level layers. The lowest layer, network-specific transport protocols, is defined in T.123 and covers also TCP/IP via a reference profile. The second layer, Multi-point Communication Service (MCS), defined in T.122 and T.125, builds upon it and adds the key feature to T.120, multipoint data delivery. The third layer is Generic Conference Control (GCC), defined in T.124. It is referenced by the Generic Application Template (GAT, T.121), which in turn is the building block for standard T.120 application protocols as well as for non-standard application protocols. The application protocols may also use the services of the MCS directly.

The Multi-Point Communication Service (MCS) is probably the most important part of the T.120 protocol stack. As already mentioned, it provides multi-point data delivery. In addition, it guarantees that any data sent to the conference participants will arrive in order. Segmentation of data is provided automatically, but the application in the receiving end must take care of the reassembly. Data may be sent with different priority levels. File transfer will have the lowest priority, while error reporting has the highest priority. Last but not least, MCS provides tokens, which can be used e.g. for coordinating the conference.

Conferences will be managed by the Generic Conference Control (GCC) layer. It provides rudimentary conference security and allows the applications to negotiate about their capabilities. In this way, the applications can find out if they can interoperate and at what feature level. GCC can also dynamically track MCS resources and prevent conflicts for them. Finally, with GCC it is possible to support the concept of conductorship in a conference.

The Generic Application Template (GAT, T.121) actually isn't any layer in the T.120 protocol stack, but it provides a template for application developers that they should use as a guide for building application protocols.

It should be noted that T.120 does not define any protocols for transferring video or audio. It defines only two standard application protocols: Still Image Exchange (T.126), for whiteboard functions, and Multi-Point File Transfer (T.127). It is probable that vendors develop proprietary, mutually incompatible protocols for other purposes. On the other hand, the existing ITU-T standards will probably be adopted for video conferencing - only its mapping to T.120 may cause incompatibility. Time will show how well T.120 will be adopted by the large audience.

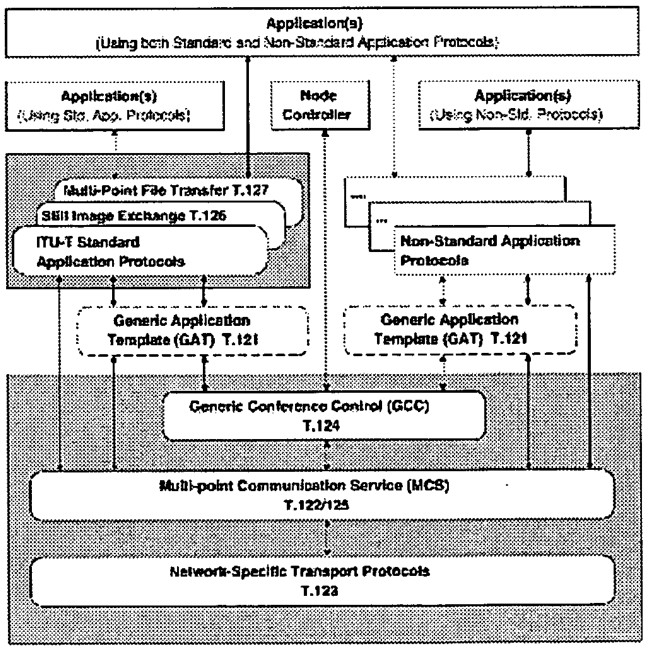
Application(s)
(Using both Standard and Non-Standard Application Protocols)
Application(s)
(Using Std. App. Protocols)
Node Controller
Application(s)
(Using Non-Std. Protocols)
Multi-Point File Transfer T.127
Still Image Exchange T.126
ITU-T Standard Application Protocols
Non-Standard Application Protocols
Generic Application Template (GAT) T.121
Generic Application Template (GAT) T.121
Generic Conference Control (GCC) T.124
Multi-point Communication Service (MCS) T.122/125
Network-Specific Transport Protocols T.123

Figure 2.3: The T.120 Protocol Stack
Reprinted with permission of DataBeam Corporation

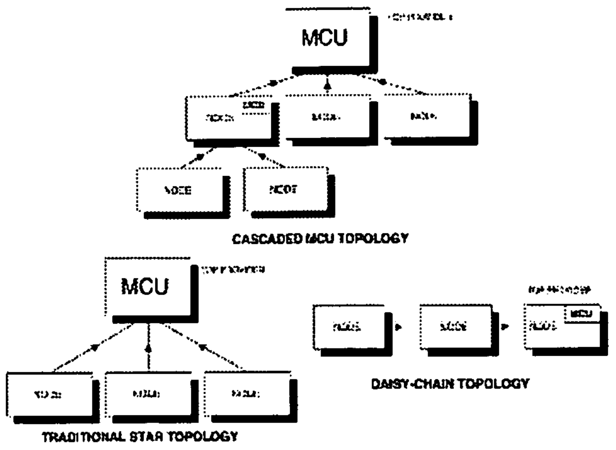

Figure 2.4: Sample T.120 Topologies
Reprinted with permission of DataBeam Corporation

2.4.2 ITU-T H.323, Video Communication on LANs

According to [12], the ITU-T standard H.323 consists of the following core components:

H.225	Specifies messages for call control including signaling, registration and admissions, and packetization/synchronization of media streams.
H.245	Specifies messages for opening and closing channels for media streams, and other commands, requests and indications.
H.261	Video codec for audiovisual services at P x 64kb/s.
H.263	Specifies a new video codec for video over POTS and low-bitrate lines.
G.711	Audio codec, 3.4 kHz at 64 kb/s (normal telephony).
G.722	Audio Codec, 7 kHz at 48, 56, and 64 kb/s.
G.728	Audio Codec, 3.1 kHz at 16 kb/s.
G.723	Audio Codec, 5.3 and 6.4 kb/s.
G.729	Audio Codec, 8 kb/s.

[12] claims that all of these components have been ratified by October 1996, and there are already some H.323-based products on the market. ITU-T standards cost money, but some H.323 related drafts are fortunately available from an FTP site. The H.225.0 version 2 draft [13] was dated March 1997, so there is still some work going on in the standardization process.

The list of the H.323 components hints that the standard can be used in very low bitrate applications as well as on fast networks, which will offer better audio and video quality.

On a side note, the Motion Pictures Expert Group (MPEG) is working on the MPEG-4 standard, which was originally planned to be a video codec for teleconferencing over very low bitrate lines. This standard will probably be finished during 1997, and time will show which codec will be adopted by the vendors, H.263 or MPEG-4. It should be noted that MPEG-4 will not merely be a video codec, but a much larger concept that contains a scripting language and other features. H.263 is a very popular video codec even in today's applications, and MPEG-4 could require too big programming efforts and computation power in order to be feasible everywhere.

2.4.3 ITU-T H.225, Media Packetization and Synchronization on LANs

The ITU-T H.225 recommendation [13] describes how audio, video, data, and control information on a non-guaranteed quality of service LAN can be managed to provide conversational services in H.323 equipment. It makes use of the Real Time Protocol (Section 2.3) for media stream packetization and synchronization for all underlying LANs, but the recommendation emphasizes that this does limit the use of H.225 to TCP/IP or Ethernet networks.

H.225 describes the communication between H.323 terminals and H.323 gateways on the same LAN, using the same transport protocol. The protocols used between the gateways are beyond the scope of the recommendation. Also, the recommendation does not include any means for assuring an adequate quality of service (QoS), except that the lack of it can be found out by using RTCP.

The H.323 gateways may perform data conversions due to different bit rate limitations. Compression and silence suppression will save bandwidth on packet-switched networks, but they are of no use on circuit-switched networks.

The recommendation defines Transport Service Access Points (TSAPs) for different purposes. Real-time channels (for distributing video or audio) as well as RAS (Registration, Admission and Status) channels are unreliable. Reliable connections are used only for call signaling, for H.245 (for opening and closing channels for media streams and for video recorder like user interface) and for T.120 data channels.

Like the whole ITU-T H.323 family, H.225 is extensible. It defines several audio coding methods and one video encoding method, H.261. Other methods can be used after a negotiation between all endpoints.

The H.225 recommendation is the glue that ties the H.323 standard together. It makes extensive use of RTP and RTCP and applies them also to other networks than the Internet, including the public switched telephone network (PSTN). With H.323, it is possible to hold video conferences with participants both from a local network and from the PSTN. If the standards get adopted by enough manufacturers and users, they will probably set a reference platform also for multimedia teleconferencing in the Internet.

2.4.4 Applications Based on the ITU-T Standards

Although the standards in this area are rather young, there are already some commercial implementations, because major software companies have belonged to the working groups.

DataBeam corporation has been very active in the T.120 standard development, and it has several products for multimedia teleconferencing. One notable product, the neT.120 Server, is the backbone for multipoint conferencing. It is available for Windows NT and Solaris. DataBeam also has a client, FarSite, which is only available for Microsoft Windows. DataBeam claims on its page that there is multimedia conferencing software also for different UNIX platforms, but did not mention any product names.

Microsoft's NetMeeting, which is part of their Internet Explorer 3.0, is another client implementation for Windows. It allows multipoint audio and video conferencing with a virtual whiteboard, and implements also the multipoint file transfer. At least some of the multipoint features require a separate MCU (Multipoint Control Unit) server, like DataBeam's neT.120, running on some other computer.

For the Apple Macintosh, there is Apple Media Conference, which has roughly the same features as DataBeam's and Microsoft's products.

2.5 Voice over Internet

Telephony requires fairly little bandwidth, but phone companies have traditionally priced especially long-distance calls quite high. Transferring speech over the Internet is much cheaper than making a long-distance call. The market is open for Internet telephony applications.

There are several commercial products that support audio over the Internet. In the survey by Pulver.com, Inc., these products were divided into two categories: Internet Telephony, which builds upon connectionless services (UDP), and Streaming Audio/Video Products, which require a reliable stream transfer protocol (TCP/IP).

According to Pulver.com's survey, the market leader in Internet Telephony is Vocaltec's Iphone. It is only available for Windows and Macintosh platforms, and there were no technical specifications on Vocaltec's site. Vocaltec also has a PSTN-to-Internet gateway that makes normal phone calls over the Internet possible.

Progressive Networks' RealAudio, published in 1995, was the first widely spread voice-over-Internet application. It still holds the first place in Point.com's list of streaming audio/video products. RealAudio and RealVideo build on a TCP/IP stack, and they are available for a wide variety of platforms. It is meant only for PC-to-PC communications. Progressive Networks did not offer any technical specifications for its products either.

It seems that the market leadership is being held by proprietary products right now. Time will show if any standard will take over. The RTSP Audio format [8] is a candidate for an open standard, but unfortunately it does not propose any concrete audio codec algorithm.

2.6 Real Time Streaming Protocol (RTSP)

The Real Time Streaming Protocol (RTSP) [8] has been developed for controlling real time data streams. The protocol is built upon Session Control Protocol (SCP), which supports multiple sessions on a single TCP stream by dividing the session streams into segments, which carry a small header that identifies the sessions. Also the Session Control Protocol has been defined in [8].

The normal SCP header is 8 bytes, but there is a compressed variant that is most efficient with small segments identified by small session numbers.

RTSP uses two SCP sessions on a single TCP stream. The actual media stream transfer can also take place on the stream, but this is not always desirable for very delay-sensitive streams that would be better distributed over a connectionless protocol. For this reason, RTSP can be used to control unicast and multicast RTP [9] data streams (Section 2.3).

2.6.1 RTSP Applications

The RTSP draft [8] gives several examples of applications where RTSP can be used. In addition to real-time live feeds such as a live Internet radio station, RTSP can be used for transferring stored real-time clips, e.g. for on-demand retrieval of recorded audio or video streams. Although RTSP was designed for real-time purposes, it can also be used for transferring non-real-time data, such as a text track related to an audio or video stream.

For transferring audio streams, the paper suggests an RTSP Audio format and a format for audio annotations. The audio format is merely a container for the audio stream. The header contains general information, such as number of channels, bits per sample, samples per second, samples per frame, and so on. Last but not least, it identifies the coder/decoder algorithm, which is needed for decoding the stream. The suggested annotation format allows for a variable amount of data fields, called chunks. They can carry copyright information, name of the audio clip, and a URL of a site where the codec can be downloaded, in case it is unknown by the recipient. Any unrecognized chunks should be ignored by the client.

RTSP is mainly a protocol for controlling real time streams, which can be implemented using other protocols. It provides mechanisms to request delivery of real-time data or information about the data, and to make a VCR-like interface possible with primitives for starting, stopping and pausing the delivery, and even for random access to some streams.

2.6.2 RTSP Messages

An RTSP session begins with the client and the server introducing themselves using the `HELLO` primitive. The server may want to authenticate the client by sending an `ID` message and by checking the `ID_RESP` message it will get in response. Several authentication methods are supported, and these two messages can be used to negotiate the authentication method.

If a resource has moved, the server will send a REDIRECT message with the new URL, just like HTTP 1.0 does for hypertext documents. The OPTIONS message can be used to negotiate various options between the server and the client. The GOODBYE message is sent when the client wants to close the connection.

The client uses the FETCH message whenever it requests a stream. In addition to the URL of the resource, the client specifies an estimate of the available bandwidth and the maximum number of proxy servers it allows between itself and the server. Flags may be used to override proxies, to indicate multicast capability, and to request unspecified bit rate (for non-real time TCP streams).

If the server can provide the requested stream with the specified parameters, it replies with a STREAM_HEADER message that gives more information about the stream: stream family (currently only RTSP Audio has been defined), bit rate, last modification time, length, maximum packet size, and some flags. The client uses this information and chooses the transport channel: UDP, SCP or compressed SCP. It may also get or set some parameters with the GET_PARAM and SET_PARAM messages, which will be acknowledged with PARAM_REPLY or PARAM_REFUSE by the server. Finally, the client will use the SET_TRANSPORT message to set the delivery mechanism (protocol and addressing parameters) for the transfer. The server will start sending data as soon as it receives a PLAY_RANGE message that specifies which range of the stream should be transferred. It will respond to the message with STREAM_SYNC.

While the stream transfer is in progress, it can be controlled with messages like SET_BLOCK_SIZE and SET_SPEED. Further video recorder like controls are made possible by the STOP and RESUME messages, and of course also by the PLAY_RANGE message. The server can get feedback and statistics from the client by sending a SEND_REPORT request, to which the client will reply with a REPORT message.

The server reports any errors to the client with an ERROR message. In addition to an error code and a fatal error flag (which specifies if the connection will be terminated), the error message is presented in plain text as well. RTP (UDP) transmission errors are handled with the UDP_RESEND message, which the client will send if it did not receive some packets correctly.

The protocol allows for custom extensions. The EVENT_NOTIFY message can be used to transfer custom messages. The currently defined payload types are NewURL and GotoURL. For both types, the payload is a text string containing the URL, but it could be anything else for some other payload type.

Most authors of the draft [8] work for Netscape Communications Corporation, so it is very likely that RTSP will have an important role in future Internet applications.

3. On-Demand Applications

On demand applications differ from real-time applications in one important aspect. Both applications typically require a guaranteed bit rate, but passive on-demand applications tolerate end-to-end delays much better than interactive real-time conferences where even a delay of half a second between the participants is considered intolerable.

Thanks to the better delay tolerance, on-demand applications can exploit large buffers to increase the tolerance against jitter in the network round-trip delay and also against dropped packets to some extent. Also, asymmetric compression algorithms may be used in video compression. MPEG-2 video compression in real time is either impossible or too expensive to be built into teleconferencing terminal equipment. In on-demand applications, the video streams have typically been recorded somewhere, i.e. they are not live broadcasts. Because of this, the streams may be compressed using powerful algorithms like MPEG-2 using more (or less) real time than the actual length of the video clip.

As with Voice over Internet applications (Section 2.5), proprietary implementations such as RealVideo by Progressive Networks are currently dominating the market, and standards are being developed or they have just come out.

Despite the differences between on-demand and real-time applications, the network and protocol requirements are quite the same for both. RTP, RTCP [9] and RTSP [8] are the standard means for controlling the transfer of audio and video streams. In high-end applications such as movies, ATM cell switching routers (Section 2.1.1) or something must bypass the bottleneck of the Internet, the routers.

4. Interactive Media

Interactivity has become a buzz-word in this decade. Applications such as WWW are possible due to few important protocols. One of them is definitely Hypertext Transfer Protocol (HTTP). The other crucial development in interactivity is Virtual Reality Modeling Language (VRML).

4.1 Hypertext Transfer Protocol

4.1.1 HTTP 1.1

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. HTTP has been in use by the World-Wide Web global information initiative since 1990. The structure of HTTP has been changed two times, first from the initial version HTTP/0.9 to HTTP/1.0, then to the current standard HTTP/1.1 [14]. HTTP/1.1 is fairly new, standardized in January 1997, and will replace HTTP/1.0 in time. The protocol allows an open-ended set of methods that indicate the purpose of a request. It builds on the discipline of reference provided by the Uniform Resource Identifier (URI), as a location (URL) or name (URN), for indicating the resource to which a method is to be applied. Messages are passed in a format similar to that used by Internet mail as defined by the Multipurpose Internet Mail Extensions (MIME).

HTTP is also used as a generic protocol for communication between user agents and proxies/gateways to other Internet systems, including those supported by the SMTP, NNTP, FTP, Gopher, and WAIS protocols. As a result, HTTP allows basic hypermedia access to resources available from diverse applications in many interactive services.

4.1.2 Secure HTTP

Secure HTTP is a secure message-oriented communications protocol designed for use in conjunction with HTTP. It is designed to coexist with HTTP's messaging model and to be easily integrated with HTTP applications. Consequently, it mimics much of HTTP's style and syntax.[15]

Secure HTTP provides a variety of security mechanisms to HTTP clients and servers, providing the security service options appropriate to the wide range of potential end uses possible for the World-Wide Web. The protocol provides symmetric capabilities to both client and server (in that equal treatment is given to both requests and replies, as well as for the preferences of both parties) while preserving the transaction model and implementation characteristics of HTTP.

Several cryptographic message format standards may be incorporated into S-HTTP clients and servers. S-HTTP supports interoperation among a variety of implementations, and is compatible with HTTP. S-HTTP aware clients can communicate with S-HTTP oblivious servers and vice-versa, although such transactions obviously would not use S-HTTP security features.

S-HTTP does not require client-side public key certificates (or public keys), as it supports symmetric key-only operation modes. This is significant because it means that spontaneous private transactions can occur without requiring individual users to have an established public key.

S-HTTP supports end-to-end secure transactions, in contrast with the original HTTP authorization mechanisms which require the client to attempt access and be denied before the security mechanism is employed. Clients may be «primed» to initiate a secure transaction (typically using information supplied in message headers); this may be used to support encryption of fill-out forms, for example. With S-HTTP, no sensitive data need ever be sent over the network in the clear.[16]

4.2 Applications

4.2.1 Newspapers, Web Magazines

Due to its reachability, the Internet have interested many newspapers publishers. The Internet broadens a whole new perspective in the distribution of information. The number of the most widespread traditional newspapers have digitalized in the last few years. In some cases, the Internet is the prime distribution channel for some newspapers. The role of online-newspapers will become even more important in the future.

The rapid spread of digital technologies and the societal shift to «being digital» reflects three compelling advantages of digital media:

1. First, digital information can be easily searched, manipulated and filtered. On-line databases and CD-ROM based reference works are especially effective in extracting needed information from large amounts of data.
2. Second, information can be delivered rapidly and at low cost in digital form.
3. Finally, the storage space requirements of digital information are minimal compared to paper-based filing systems. Moreover, with a well-designed database system, the documents are more likely to be found again at least efforts compared to paper documents.

4.2.2 Educational Materials

Ever since the Internet was introduced, it has always played an important role as an information channel. It encompasses roughly 1.4 million computers with Internet addresses that are used by up to 30 million people in more than fifty countries. As more and more colleges, universities, schools, companies, and private citizens connect to the Internet either through affiliations with regional not-for-profit networks or by subscribing to information services provided by for-profit companies, more possibilities are opened for distance educators to overcome time and distance to reach students.

With access to the Internet, distance educators and their students could use:

Electronic mail (e-mail)
: Like postal mail, e-mail is used to exchange messages or other information with people. Instead of being delivered by the postal service to a postal address, e-mail is delivered by Internet software through a computer network to a computer address.

Bulletin boards
: Many bulletin boards can be accessed through the Internet. Two common public bulletin boards on the Internet are USENET and LISTSERV. USENET is a collection of thousands of topically organized newsgroups, covering everything from supercomputer design to bungee cord jumping, and ranging in distribution from the whole world to single institutions. LISTSERV also provides discussion forums on a variety of topics broken out by topic or area of special interest.

World-Wide Web (WWW)
: The WWW provides Internet users with a uniform and convenient means of accessing the wide variety of resources (pictures, text, data, sound, video) available on the Internet. Popular software interfaces, such as Netscape, facilitate navigation and use of the WWW. The central organizing feature of the WWW is the home page.

4.3 VRML

The Virtual Reality Modeling Language (VRML) is a file format for describing 3D interactive worlds and objects. VRML was conceived in the spring of 1994 at the first annual World Wide Web Conference in Geneva, Switzerland.

Shortly after the Geneva BOF session, the www-vrml mailing list was created to discuss the development of a specification for the first version of VRML [17]. A rough draft version of the specification was compeleted by the

WWW Fall 1994 conference.

VRML may be used in conjunction with the World Wide Web. It may be used to create three-dimensional representations of complex scenes such as illustrations, product definition and virtual reality presentations.

Some of the virtual reality characteristics of VRML are:

Interaction
: The sensor nodes set off events when the user moves in certain areas of a world and when he or she clicks certain objects. These nodes let the user drag objects and controls from one place to another. Another kind of sensor would keep track of the passage of time, for instance.
Collision detection ensures that solid objects react like solid objects; the objects moves by the user bounces off them (or simply stop moving) when it runs into them.

Animation
: New animation objects determined in VRML 2.0 also include animation object Interpolators. These objects allow the user to create pre-defined animations of a many aspects of the world and then play it at some opportune time. With animation interpolators you can create moving objects such as flying birds, automatically opening doors, or walking robots, objects that change color as they move, such as the sun, objects that morph their geometry from one shape to another, and you can create guided tours that automatically move the user along a predefined path.

Scripting
: VRML 2.0 wouldn't be able to move without the new Script nodes. Using Scripts, the user can not only animate creatures and objects in a world, but give them a semblance of intelligence. Animated dogs can fetch newspapers or frisbees; clock hands can move; birds can fly; robots can juggle.

Whether the final goal is educational, commercial, or technical, most compelling VRML have certain characteristics in common:

- The user enters the 3D world on the computer screen and explores it as he or she would explore part of the real world. Each person can chart a different course through this world.
- The local browser allows ther user to explore the VRML world in any way he or she decides. The computer doesn't provide a fixed set of choices or prescribe which path to follow, although the VRML author can suggest recommendations. The possibilities are unlimited.
- Objects in the world can respond to one another and to external events caused by the user. The user can reach into the scene and change elements in it.

4.3.1 Design Criteria

VRML has been designed to fulfill the following requirements:

Authorability
: Make it possible to develop application generators and editors, as well as to import data from other industrial formats.

Completeness
: Provide all information necessary for implementation and address a complete feature set for wide industry acceptance.

Composability
: The ability to use elements of VRML in combination and thus allow re-usability.

Extensibility
: The ability to add new elements.

Implementability
: Capable of implementation on a wide range of systems.

Multi-user potential
: Should not preclude the implementation of multi-user environments.

Orthogonality

The elements of VRML should be independent of each other, or any dependencies should be structured and well defined.

Performance
: The elements should be designed with the emphasis on interactive performance on a variety of computing platforms.

Scalability
: The elements of VRML should be designed for infinitely large compositions.

Standard practice
: Only those elements that reflect existing practice, that are necessary to support existing practice, or that are necessary to support proposed standards should be standardized.

Well-structured
: An element should have a well-defined interface and a simply stated unconditional purpose. Multipurpose elements and side effects should be avoided. [17]

4.3.2 Scripting Language Basics

VRML can be said to be just a way for objects to read and write themselves. Theoretically, the objects can contain anything: 3D geometry, MIDI data, JPEG images, anything. VRML defines a set of objects useful for doing 3D graphics. These objects are called Nodes.

VRML defines several different classes of nodes. Most of the nodes can be classified into one of three categories; `shape`, `property` or `group`. `Shape` nodes define the geometry in the world. Conceptually, they are the only nodes that draw anything. `Property` nodes affect the way shapes are drawn. And `grouping` nodes gather other nodes together, allowing collections of nodes to be treated as a single object. Some group nodes also control whether or not their children are drawn.

Below is a file which contains a simple scene defining a view of a red sphere and a blue box, lit by a directional light:

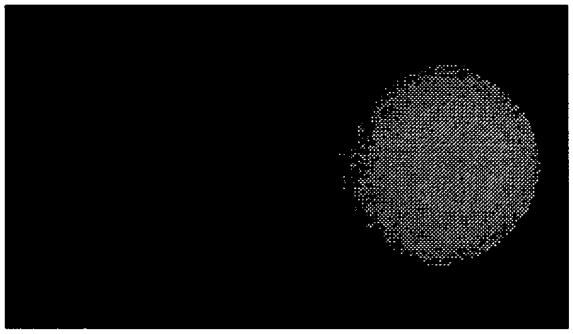

Figure 4.1: VRML Sample
Reprinted from VRML version 2.0

```
#VRML V2.0 utf8
Transform {
  children [
    NavigationInfo { headlight FALSE } # We'll add our own light

    DirectionalLight {        # First child
        direction 0 0 -1      # Light illuminating the scene
    }

    Transform {               # Second child - a red sphere
      translation 3 0 1
      children [
        Shape {
          geometry Sphere { radius 2.3 }
```

```
            appearance Appearance {
              material Material { diffuseColor 1 0 0 }   # Red
            }
          }
        ]
      }

      Transform {                 # Third child - a blue box
        translation -2.4 .2 1
        rotation     0 1 1  .9
        children [
          Shape {
            geometry Box {}
            appearance Appearance {
              material Material { diffuseColor 0 0 1 }  # Blue
            }
          }
        ]
      }

    ] # end of children for world
  }
```

However, VRML does not directly determine any scripting language, but rather it has recommedations for JavaScript and Java. The JavaScripts are confined certain browsers and thus they can not be used in every case. Therefore, the Java-interface is almost the best language for VRML.

5. Traditional vs. Digital Media

When comparing both traditional and digital media, the disctinction of the two should be made.

- Traditional Media
  - Newspapers
  - Radio
  - Television
- Digital Media
  - WWW
  - Newsgroups
  - VIDEO Applications

5.1 Advantages of Digital Media

Digital media has a few advantages over the traditional ones:

Comprehesiveness
: No physical library can compare to the mountains of information available on a stack of CD-ROMs or on the Internet. Of course the comprehensiveness goes hand in hand with the next advantage.

Searchability
: Commanding the computer to search through comprehensive database is a major editorial advantage over print, useful for retrieving a nugget of data but more importantly helping the reader sift and filter raw data into useful knowledge.

Economy
: Except for way-out-off-hand multimedia projects with too much new video and overly-complex programming, digital media is always cheaper and faster to produce than traditional media.

Transaction
: The most important new media advantage, the transaction feature lets the reader talk back through a newsgroup,

order directly from an on-line catalog or buy books, software and music off a CD-ROM.

5.2 Drawbacks of Digital Media

While these advantages are compelling, the construction of a truly effective and functional digital media cannot fully be implemented yet. Massive investments will be required to upgrade existing networks as well as build new ones with broadband capabilities. The risks of such investments are huge given that no one is sure exactly what the product will be, how to charge for it, and how much consumers might be willing to pay.

Moreover, rapidly changing technological standards make consumers, as well as communications firms, reluctant to commit to a particular technology.

The impermanence of magnetic storage media, as well as changing standards, provide little assurance that today's information will be intact or even accessible in 20 years time. To ease the unity of digital media, the standard recommended should be followed.

Finally, access to the information highway is limited to those with a piece of expensive electronic equipment connected to the network. Even with the development of the few thousands FIM network computer, most of the world's population will be effectively excluded from tapping into the network, even in the developed countries.

6. Legal Issues

6.1 Telecommunications Monopolies

The Finnish telecommunications market is perhaps the most liberalised in the whole of Europe. In Finland, telecommunications operations have been opened up gradually to competition since 1985. The liberalisation process has not been as difficult in Finland as in many other countries because part of the telecommunications operators has always been privately owned. At the moment there are about 70 telecommunications operators in Finland. [18]

This has had a positive impact on telecommunications in Finland [19]. Since 1987 the Telecommunications Act [20] has removed all the barriers created by monopolies. However, most of the teleoperators have found their own specific markets.

6.2 Copyright Issues

Along with the reachability and accessibility of the digital media in the Internet issues such as copyright issues need to be considered. Once the material is exposed to public, it is difficult to control the abuse of the material. The problem depends on how the «Fair Use» [21] of the material is defined. The Fair Use is initially started by the active intellectual property authorities in the United States. Here in Finland conventions pertaining to copyright issues are not yet fully completed, although some preliminary efforts have been made.

7. Future Aspects

It is inevitable that the Internet will serve an important role in transferring different applications of digital media. As technology advances more applications will become more standardized and further developped. In the constantly changing information high-way, it is impossible to anticipate the exact changes. However, it is only possible to predict a certain trend in development of digital media. As the Internet become reachable and affordable to most people in the world, the future of digital media is unlimited.

Growing demand on applications and numbers of users also increases the demand of security in distribution digital media in the Internet. The current hypertext transfer protocol will be modified and its security will be examined many times in the years to come. In fact, confidentiality and reliability between the endpoints, whether they are real persons or not, are the most crucial factors in terms of the development of digital media.

VRML will be in a wider use. Especially the electronic commerce has been very interested in the various VRML applications. Customers will be able to see a 3D visualization of the product. VRML can be used in planning and modeling new items as a help tool for CAD and so forth.

8. Conclusion

Digital media is one of the most rapidly developing topics in information technology. In the past ten years, the changes have been tremendous. Much of the development would have been difficult to expect or to predict ten years ago. In order to make the applications and development efforts work together, standards must be written.

The continuously growing number of Internet users and the increasing popularity and use of real-time applications place great demands on the infrastructure, and the capacity of the backbones must be improved in many countries in order to make response times acceptable, or the Internet will divide into two. Only users in advanced countries, such as Finland, will be able to utilize the new bandwidth-hungry applications, while users in the «developing countries» must cope with e-mail and other low-bandwidth services.

The much abused connection-oriented data circuits such as the PSTN have one advantage over the Internet: quality of service guarantees. But the situation will hopefully change if the backbones will be built with ATM and if traditional routers will be replaced with cell-switching ones.

One of the most exciting characteristics of the digital media is its interactivity. Online newspapers and Web magazines have become a quick and inexpensive information source for many people. Another application that benefits from interactivity is distance learning. Perhaps the most interesting interactive language is the Virtual Reality Modeling Language (VRML), which could become the engine for modern educational, commercial and technical applications.

Although digital media has a growing importance in our lives, the traditional forms of media cannot be ignored. New interactive media applications are more likely to be accepted if there is an equivalent form of traditional media. Moreover, despite many positive characteristics of digital media, it also lacks some qualities that only traditional media can possess. Printed text is easier to read than text on a computer screen, and you can take a piece of paper everywhere with you.

Many countries still lack competition in the telecommunications sector, and even in countries that have enabled competition years ago, teleoperators have local monopoly in most parts of the country. There is a great danger that teleoperators want to make money with obsolete technology such as ISDN and will not build better infrastructure, because they fear that it would be outdated in a few years anyway. True competition will only be possible in the biggest cities.

Distributing digital media in the Internet is not as simple as it seems on the first sight. It involves controversial issues like financial problems and copyright. Many questions arise. Who builds and pays the infrastructure? Who controls the distribution of information in the Internet? Need the copyright conventions be changed because of Internet, and to which direction? To what extent can the materials found in the Internet be used in other works?

It is obvious that new protocols and technologies will be introduced and modified along with the demand and changes. Everyone is willing to standardize their own techniques. Standards must be developed rapidly, or proprietary de-facto standards will conquer the market. Standardization groups must be open for developers and vendors, or otherwise their decisions will not be listened to. ISO and ITU-T are slowly learning their lesson, and the disputes between the Internet community and these traditional standardization organs are being buried.

Abbreviations

ABR	Available Bit Rate (in ATM)

ATM	Asynchronous Transfer Mode
CAD	Computer Aided Design
CBR	Constant Bit Rate (in ATM)
CSCW	Computer Supported Collaborative Work
CSR	Cell Switching Router (ATM switch+IP router)
CSRC	Contributing Source (in RTP, e.g. a mixer)
FTP	File Transfer Protocol
GAT	Generic Application Template (T.120)
GCC	Generic Conference Control (T.120)
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
IRC	Internet Relay Chat, textual real-time many-to-many communications
ITU	International Telecommunications Union
ITU-T	Telecommunication Standardization Sector of ITU
ISDN	Integrated Services Digital Network (PSTN)
JPEG	Joint Picture Expert Group, a compression method of still images
LAN	Local Area Network
MARS	Multicast Address Resolution Server (in ATM)
MBone	Internet Multicast Backbone
MCS	Multicast Server (in ATM). See also MARS.
MCS	Multipoint Communication Service (in T.120)
MCU	Multipoint Control Unit (in T.120)
MIDI	Musical Instrument Digital Interface, also a format for coding music events
MIME	Multipurpose Internet Mail Extensions
MPEG	Motion Pictures Expert Group, compression methods for video
NNTP	(Use)Net News Transfer Protocol
PC	Personal Computer
POTS	Plain Old Telephony Services (nickname for PSTN)
PSTN	Public Switched Telephone Network
QoS	Quality of Service
RAS	Registration, Admission and Status [channel] (H.225)
RSVP	Resource Reservation Protocol
RTP	Real Time Protocol, builds on UDP and IP multicasting
RTCP	RTP Control Protocol
RTSP	Real Time Streaming Protocol
SAP	Session Announcement Protocol
SCP	Session Control Protocol (TCP substream, defined in RTSP)
SDP	Session Description Protocol
SMTP	Simple Mail Transfer Protocol
SSRC	Synchronization Source (in RTP, e.g. a PC)
TCP/IP	Transmission Control Protocol/Internet Protocol, a connection-orientated end-to-end service
TSAP	Transport Service Access Point

TTL	Time To Live (routing hops of IP packets)
URI	Uniform Resource Identifier (WWW)
URL	Uniform Resource Locator (WWW)
URN	Uniform Resource Name (WWW)
VBR	Variable Bit Rate (in ATM)
VC	Virtual Circuit (in ATM)
VP	Virtual Path (in ATM)
VRML	Virtual Reality Modelling Language
UBR	Unspecified Bit Rate (in ATM)
UDP	User Datagram Protocol, connectionless packet delivery on IP
WAIS	Wide Area Information Services
WWW	World Wide Web

Refererences

Many of the documents are Internet-Drafts, working documents of the Internet Engineering task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as «work in progress».

To learn the current status of any Internet-Draft, please check the «1id-abstract.txt» listing contained in the Internet-Drafts Shadow Directories on ftp.is.co.za (Africa), nic.nordu.net (Europe), munnari.oz.au (Pacific Rim), ds.internic.net (US East Coast), or ftp.isi.edu (US West Coast).

[1] H. Esaki. Router Architecture Extensions for ATM: Overview. *Internet-Draft*, <draft-rfced-info-katsube-00.txt>, November 1996.

[2] T. Smith, G. Armitage. IP Broadcast over ATM Networks. *Internet-Draft*, <draft-ietf-ion-bcast-01.txt>, November 1996.

[3] M. Garrett, M. Borden. Interoperation of Controlled-Load and Guaranteed-Service with ATM. *Internet-Draft*, <draft-ietf-issll-atm-mapping-01.txt>, November 1996.

[4] C. Semeria, T. Maufer. Introduction to IP Multicast Routing. *Internet-Draft*, <draft-ietf-mboned-intro-multicast-00.txt>, January 1997.

[5] M. Handley. SAP: Session Announcement Protocol. *Internet-Draft*, <draft-ietf-mmusic-sap-00.txt>, November 1996.

[6] M. Handley, V. Jacobson. SDP: Session Description Protocol. *Internet-Draft*, <draft-ietf-mmusic-sdp-03.txt>, November 1996.

[7] A Primer on the T.120 Standards. *DataBeam Corporation*, <URL:http://www.databeam.com/Products/CCTS/t120primer/t120primer.html>.

[8] A. Rao, R. Lanphier. Real Time Streaming Protocol (RTSP). *Internet-Draft*, <draft-rao-rtsp-00.txt>, October 1996.

[9] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. *Internet Request for Comments*, RFC 1889, January 1996.

[10] B. Aboba. Payload Format for HTTP Encoding in RTP. *Internet-Draft*, <draft-aboba-rtp-http-02.txt>, January 1997.

[11] C. Zhu. RTP Payload Format for H.263 Video Stream. *Internet-Draft*, <draft-ietf-avt-rtp-payload-02.txt>, November 1996.

[12] H.323 ITU Standards. *The International Multimedia Teleconferencing Consortium*, <URL:http://www.imtc.org/i/standard/itu/i_h323.htm>.

[13] Draft ITU-T Recommendation H.225.0, Version 2. *Telecommunication Standardization Sector of International Telecommunication Union*, <URL:ftp://itu-t:sg15!avc@ftp.gctech.co.jp/9703_Gen/h225v205.doc>, March 1997.

[14] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, T. Berners-Lee. Hypertext Transfer Protocol -- HTTP/1.1, *Internet Request for Comments*, RFC 2068, January 1997

[15] E. Rescorla, A. Schiffman. The Secure HyperText Transfer Protocol. *Internet-Draft*, <draft-ietf-wts-shttp-04.txt>, March 1997.

[16] K. Holtman, A. Mutz. Transparent Content Negotiation in HTTP. *Internet-Draft*, <draft-ietf-http-negotiation-01.txt>, March 1997.

[17] The Virtual Reality Modeling Language Specification Version 2.0. *ISO/IEC CD 14772*, <URL:http://irdu.nus.sg/vrml/specification/v2.0final/>, August 4, 1996.

[18] Operators for Public Telecommunications. *Telecommunications Unit, Ministry of Transport and Communications, Finland*, <URL:http://www.vn.fi/lm/vho/tu/tcs.htm>, November 1996.

[19] Televiestintä Suomessa. *Ministry of Transport and Communication, Finland*, <URL:http://www.vn.fi/lm/vho/televies.htm>, April 1997.

[20] Telecommunications Act. *Communications Administration Department*, <URL:http://www.vn.fi/lm/vho/tu/act.htm>, February 1987.

[21] Copyright & Fair Use. *Stanford University Libraries*, <URL:http://fairuse.stanford.edu/>, January, 1997.

Date: Mon 21 Mar 94 05:51:23 PST
From: Stephen Casner <CASNER@ISI.EDU>
Subject: AVT WG Agenda and Proposed RTP Changes
To: rem-conf@es.net

To the Audio/Video Transport WG:

You may be wondering about the status of AVT in general and of the Real-time Transport Protocol (RTP) in particular.

Just before the November IETF meeting in Houston, the RTP spec was submitted to the Area Director with a request for "IESG Last Call". At the AVT meeting, comments were solicited on the spec but none were offered. However, behind the scenes, some objections were raised with the Area Directorate regarding the classification of RTP as a proposed standard and some aspects of the specification. I have received some feedback on the Area Directorate's review of RTP through discussions with Allison Mankin and John Wroclawski, but I have not received a written response yet. I was hoping to get that first, but I'll try to cover the issues with this message anyway.

Subsequently, I've had two discussions with Van Jacobson and Ron Frederick in which we've taken the vat and nv programs as examples against which to test the issues listed below. As a further verification, Van and Ron plan to implement test versions of vat and nv incorporating RTP with the changes proposed below (there are hopes for this to be done before IETF). Please read about the proposed changes below, and if you can send comments before IETF, it would be much appreciated. This will be the main topic for Seattle.

The current released RTP draft is draft-ietf-avt-rtp-04.txt and .ps even though its listed expiration date is past. There is a newer version on gaia.cs.umass.edu, in pub/hgschulz/rtp, files rtp.txt and profile.txt with updates for the timestamp change proposed below.

-- Steve Casner

Audio/Video Transport WG

A G E N D A

Tuesday, March 29, 1330-1530

- Status of RTP and the working group

- Presentation on changes proposed in response to Area Directorate review:

  1. Control and data on separate ports
  2. Remove Channel ID, put multiplexing into encapsulations
  3. Replace data options with a bit field
  4. Media-specific timestamps
  5. Application-specific sync marker bit
  6. Global source identifiers
  7. Control packet format / Reception reports

- Report on test implementations of proposed changes

- Discussion of proposed changes -- what issues and problems may have been overlooked?

Wednesday, March 30, 1330-1530

- Continuation of discussion of proposed changes as needed

- Presentation on RTP and Synchronization Protocol

- Presentation of new encoding specifications over RTP (may include: JPEG; MPEG1/MPEG2; Cell-B; new version of H.261)

- Assess schedule for RTP completion and other work needed

Proposed Changes to RTP

My take on the AVT process is that we have designed a protocol that is well specified and certainly functional if not optimal. However, part of the criticism was that RTP follows a "classical" protocol style rather than the principles of Application Level Framing (ALF) and Integrated Layer Processing (ILP) proposed by David Clark and David Tennenhouse in their SIGCOMM '90 paper. I have some trouble with this claim because I believe RTP already adheres to at least the major points of these principles. For example, each packet is typically one application data unit and includes the identification required to enable processing of application data units out of order. So, the fit is really a matter of degree. The following guidelines are relevant:

- Minimize the number of multiplexing points
- Minimize the number of inband control operations

As applied to RTP, the following changes are proposed (though not all of these are exactly motivated by ALF):

- Carry the control and data traffic on separate ports
- Remove the application-level multiplexing of the Channel ID and move it to an encapsulation for the cases where it's needed
- Minimize the use of options
- Make the definition of some fields application-specific (in particular, the timestamp clock rate and sync marker)
- Use global rather than local IDs, to be able to detect loops
- Specify more precisely how reception reports should be provided

Each of these items is described in more detail below.

1. Control and data on separate ports

This change removes RTCP functions from RTP data packets and puts them into a separate packet stream on another port to streamline data processing and to allow monitoring programs to receive the control information with having to sort through the data. The relationship of port numbers does not have to be even/odd pair, only some arithmetic algorithm (such as +1) so that the mapping can be calculated in either direction. This is so that a set of ports can be identified as being in use when either control or data traffic is observed.

Advantages:

- Allows monitors to collect feedback from receivers without having to sort through the data packets.
- Moves multiplexing of control and data from option decoding to the exising multiplexing point of port number to streamline data processing.
- Enables elimination of options in data packets.

Disadvantages:

- Consumes more port number space.
- It may be harder to convince some firewall administrators to allow multiple ports through the firewall than one (although this becomes moot if port numbers are randomly allocated).

2. Remove Channel ID, put multiplexing into encapsulations as needed

There was quite a bit of debate about the Channel ID, with Christian Huitema being notable among those who opposed it. ILP says the number of multiplexing points should be minimized, which argues for removal of the Channel ID. The argument for the Channel ID was not strong, but two cases were identified where the Channel ID was needed:

- When multiple RTP units are carried in one UDP packet (to reduce packet overhead); this is indicated by a profile that specifies an additional encapsulation that currently includes only a frame length. The frame length is 32 bits for purposes of alignment, but really only 16 are needed. The other 16 bits could carry a demultiplexing field to take the place of the Channel ID.

- For some applications, the multiplexing of the Channel ID might be sufficient without UDP, so RTP could be carried directly over IP. Without the Channel ID, such applications would have to use UDP, or a new encapsulation (perhaps 32 bits) could be defined.

Advantages:

- Eliminates a multiplexing point.
- Frees bits for indication of optional fields.

Disadvantages:

- No multiplexing provided for RTP-over-IP case.

3. Replace data options with a bit field

With the RTCP operations separated into another packet stream on a different port, there are only a few optional fields to be selected in data packets. That allows the optional fields to be indicated by bits in the fixed header. If the bits are also grouped together, it is possible to interpret them as a "packet format", which is a single demultiplexing point for processing as a set of fixed formats if appropriate. (If the options were not orthogonal, a more compact encoding of the packet format would be possible.) Or, rather than replicating code, the option bits may be processed individually, which is more likely for the existing applications. The order of the bits and the corresponding fields is fixed, but the receiver may process them in any order. This reduces the validation required.

The RTP options would be replaced as follows:

CSRC (content source) -- A bit field of 5 or 6 bits in the fixed header to count the number of content source identifiers to follow, for packets produced by a bridge (mixer). Zero indicates that the packet did not come from a bridge. This count takes the place of the length field in the current CSRC option.

SSRC (synchronization source) -- One bit to indicate that a sync source identifier follows.

The CSRC count and SSRC bit could be combined into a "source ID" count if the first source ID in the list is defined to identify the sync source. That is, a packet produced by a bridge would always have to identify the bridge explicitly as the sync source, so the count would be at least 2, for the sync source ID and one content source ID. This extra overhead of the explicit sync source ID may be justified if global identifiers are used (see below) and the implicit source ID is reserved for traffic generated by the bridge host itself.

BOS (beginning of sync unit) -- This option carries the sequence number of the first packet of a synchronization unit. It has not been used yet, and it would be eliminated as a generic RTP option. A profile could define this field to be included after the RTP header.

APP (application-specific) -- Application specific functions could be defined in profiles as extensions to the RTP header, but there is no provision for one implementation of an application to define its own optional information in a way that other implementations can simply ignore.

SDST (sync destination, or reverst-path option) -- The same mechanism that is used for the SSRC on a forward packet can be used for SDST on a reverse path packet because the directions are distinguished by the arrival port (the current SSRC and SDST options could have been one). However, if global identifiers are used, it may not be possible to implement reverse path packets (some would say that's a good thing). This is because translators would not need to keep a table that would allow mapping source identifiers back into transport addresses.

ENC (encryption) -- Two methods:

A. When encryption is used, the whole RTP unit is encrypted (all of the RTP header and data). The receiver depends upon header validity checks (version, format, sequence number, and timestamp having reasonable values) to discard packets that should have been decrypted (or decrypted with a different key). There is no provision for an explicit initialization vector; instead zero would be used with random initial values for the sequence number and timestamp to deter a known plaintext attack, or a shared secret could be used. Since it would not be possible for a translator to insert or modify the SSRC field, the SSRC would always have to be inserted before encryption, and the local identifier scheme would not work. The key and encryption algorithm would be specified by out of band information; key switching can be done by trying the possible keys one at a time to decrypt the header and make the validity checks.

B. The fixed portion of the RTP header (64 bits) and the SSRC field (if present) would not be encrypted in order to allow translators to insert or modify the SSRC field (so this would reduce header size in the normal case, and would work with local identifiers). One bit in the header would indicate that the packet was encrypted. As with the current RTP ENC option, the initialization vector could either be the first 64 bits of the RTP header, or an explicit value generated randomly for each packet, but the choice would be specified only by the out-of-band information. In the case of the explict IV, the 64-bit field would be inserted after the SSRC (if present), and encryption would begin after the IV field.

MIC (authentication) -- Two methods:

A. An authenticated packet would be indicated by a bit in the header which would indicate the presence of an authentication field later in the header. To work with locally unique identifiers, which must be updated when a packet goes through a translator, the SSRC field would have to be excluded from the authentication; this means it could be faked by a translator. With the global identifier scheme, the SSRC could be authenticated to have been set by one of the sources (but could still be false). For either scheme, the SSRC field must always be included in case the packet has to go through a translator, or alternatively the SSRC flag bit could be excluded from the authentication. The authentication method (covered by ENC, keyed, symmetrically encrypted, or asymmetrically encrypted), algorithm and key (if any) would be known from out-of-band information. As with the method (A) for encryption, key changes could be accomplished by trying with old an new keys in succession. Alternatively, a key descriptor could be included at the start of the authentication field. To allow some receivers to ignore the authentication without knowing the out-of-band information, a length field would be needed at the start of the authentication field.

B. Since authentication may in the future be provided at a layer below RTP, it would be advantageous if no bits were wasted once authentication within RTP became unused. This could be accomplished by using a separate encapsulation header prepended to the RTP header and distinguished from the RTP header by a different version number or some combination of bits in the first word that was sufficiently unique from a valid RTP header. One possibility would be version 0, which would otherwise indicate vat protocol, but with unused flag bits set. The first word of the authentication encapsulation should include a length field so receivers that didn't care about authentication could skip it.

The other fields in the first 32 bits of the RTP header are the version number, format, sync marker, and sequence number. It is proposed that the version number be bumped to 2 if these proposed changes are adopted. In that case, the version field could be reduced from two bits to one if we were willing to sacrifice the possibility of defining a version 3. The format field would remain unchanged. The sync marker definition might change (see below), but would remain one bit. The sequence number should stay at 16 bits for arithmetic convenience, but could be trimmed if necessary. So, the resulting header bit count would be:

min	max	
5	6	for CSRC count
0	1	for SSRC present
0	1	for encryption
0	1	for authentication

```
  1   2  for version
  6   6  for format
  1   1  for sync marker
  8  16  for sequence number
 --- ---
 21  34  (so 2 must be trimmed)
```

Advantages:

- Options may be processed collectively as a packet format, although that seems unlikely for the options defined here.
- Option bit field takes less space than the option code + length format (though longer global ID's would consume some of the space).
- More streamlined data processing, though the difference may not be noticeable compared to the data manipulation in existing apps.
- Allows processing of SSRC first, which was identified as a problem with current scheme unless SSRC option was required to be first.

Disadvantages:

- There may not be any spare bits for new options, but it may be argued that there just aren't that many variations that must be accommodated in the common part of supporting communication across a packet net (vs. application or media specific details which go later in the packet)
- Encryption and authentication key changes can't be indicated explicitly.

4. Media-specific timestamps

A more careful analysis of the technique given in the RTP spec for conversion between RTP timestamps and sample clocks has identified two problems:

- It is possible to construct unusual pathological cases that result in an off-by-1 error due to floating point arithmetic, so we can't claim it's always accurate even though it would probably work fine for all the real examples.

- The example code in the spec is insufficient because it does not point out that the received 32-bit RTP timestamp must always be extended with the high 16 bits of the NTP timestamp in order to accurately reconstruct a 32-bit sample counter. This requires that the sender and receiver both know roughly what time it is, and we believe it is not reasonable to establish that as a requirement for all uses of the RTP protocol, even though it might be a reasonable requirement within a given profile.

Also, the fact that the RTP timestamp uses the middle bits of an NTP timestamp has led some implementations to "ask the system" for an NTP timestamp as each packet is being prepared. This technique will produce timestamps with too much jitter to be valid for audio samples. Furthermore, the sampling instant, rather than the time of packet preparation, is what's really needed for synchronization.

Therefore, it is proposed that the rate at which the clock ticks, instead of always being 65536Hz, would become a parameter of the format. For some formats, such as the variable frame rate video we are now using, it may make sense to retain the 65536Hz rate, while for most audio formats, the clock rate would be set to be the same as the sampling rate, as is done for the timestamp in vat. For predefined format codes, the clock rate would be defined in the profile spec. For dynamically defined format codes, it is proposed that the clock rate be specified as a reduced rational number, with a 32-bit numerator and a 32-bit denominator. Assuming that all sampling rates are rational, this would be exact and would avoid the complications of incompatible floating point formats.

A second aspect of the current RTP timestamp is that senders with synchronized time sources (e.g., using NTP) are supposed to periodically adjust the timestamp calculation to correct for drift between the sampling oscillator and the nominal sampling rate. This is to aid in intermedia synchronization, in particular for sources from multiple sites. In the new timestamp scheme, it would still be possible to have the sample counter be relative to the same t0 as NTP and periodically adjust it at the sender to correct for drift if the sender knows the time of day. By conveying the relationship of media timing to real time in the timestamp itself, no separate explicit communication of the relationship would be needed.

However, independent of what clock rate is chosen for the timestamp, receivers would need some indication whether or not the senders knew the time of day (and how accurately) to determine whether they could use the timestamp to synchronize. If a periodic indication would be required anyway, then one could just as well communicate the relationship to real time periodically so the receiver can make the necessary adjustments (since it must be adaptive anyway).

This would remove the requirement for the sender to make adjustments, which is an advantage for continuous transmission. Adjustments to the sample clock may cause audible glitches. If the sender must adjust for drift between the sampling clock and real time, and the receiver must adjust for drift between real time and the playout clock, the number of adjustments (and glitches) may be more than twice as many as if the receiver just made adjustments for drift between the sampling clock and the playout clock. (In the extreme case, if the sampling clock and playout clock happened to be exactly the same but both skewed with respect to real time, then synchronizing with real time would require adjustments at both sender and receiver, whereas not synchronizing to real time would require no adjustments).

Therefore, it is also proposed to remove the requirement for senders to make timestamp adjustments. For example, the timestamps would just follow the input device's sample counter. For senders that do know the time of day, control packets would carry both an RTP timestamp (sample clock) and the corresponding full 64-bit NTP timestamp to establish the relationship.

Specific applications such as the talking clocks might choose to still initialize the sample counter timestamp relative to t0 and adjust periodically keep the timestamps synchronized to real time since the data is artificially generated anyway.

Advantages

- Avoids time conversion calculations at the sender and receiver, and the potential errors at the receiver.
- Doesn't require sender to know what time it is.
- Reduced number of drift corrections for continuous transmission.

Disadvantages

- Relationship to real time must be communicated periodically.
- Monitoring and recording tools would need to know about the predefined formats and their clock rates, whereas before they could be format-independent.
- A receiver must wait for the periodic control packet before synchronization can be done. In particular, a recorder would need to back calculate the real time corresponding to the first packet of a recording to know how to play back the first packets in sync with another medium, for example.

5. Application-specific sync marker bit

In the RTP specification, the sync bit was chosen to mark the last packet of a synchronization unit because that allows the receiver to also determine the first packet of the next talkspurt by its sequence number. If the sync bit marked the first packet of a synchronization unit, it would not be possible in real time to establish the end of the previous synchronization unit.

However, some applications of RTP may only care about the start of a synchronization unit. For them to successfully determine the first packet of a synchronization unit requires that both the last packet of the previous unit and the first packet of the next one be received intact. This is a disadvantage, so those applications would prefer to mark the first packet.

If you think about it, it's not necessary for the main RTP specification to define the meaning of the sync marker because there is no generic processing of that bit to be done for all applications. In fact, within a given application, it might make sense for the definition of the marker bit to be specific to a particular encoding. Therefore, it might be claimed that the sync marker does not belong in the common RTP header at all, but there are a couple of reasons for it to remain:

- Most applications will want to mark something, and to allocate a bit elsewhere may require allocating 32. Since we don't want to make the header any longer than necessary, we should try to provide that space in the common header.

- The marker bit may be useful for media-independent monitoring because loss may often occur in some relationship to the marker. It may be possible to draw some useful information from that relationship without knowing the specific definition of the marker.

6. Global source identifiers

RTP currently uses locally unique source 16-bit identifiers to keep track of distinct sources when packets flow through translators (when the original transport address is replaced by that of the translator) and bridges (when packets from multiple sources are mixed). These identifiers have the advantage of small size, but there are some disadvantages as well:

- Translators and bridges must maintain a table of incoming streams (identified by transport addresss and possibly an incoming SSRC identifier) to be mapped to outgoing identifiers.
- If a translator or bridge went down, the mapping would probably be lost, which means downstream receivers would be temporarily confused about sources.
- Local identifiers can't be used to detect a delivery loop. For example, if there is a translator from multicast to unicast followed by a translator from unicast to multicast, and somehow the two multicast trees get hooked together, a loop may form.

Earlier in the RTP design process, truly globally unique identifiers were considered. These were constructed with [type,length,value] structures to use unique network addresses of various forms. This idea was rejected because these identifiers are too large.

Van Jacobson proposes a point in between with the scheme used by vat and wb. This scheme uses 32-bit identifiers unique within a particular medium in a particular session. In other words, one site (source) may use the same identifier for each of several media in a session. In the current Internet, the 32-bit identifiers may be derived simply by using the IPv4 address of the host if there is only one source per host (as is common for workstations), or otherwise chosen at random from the "class F" IP address space (26 bits).

There are some obvious difficulties in this scheme:

- Two sources may choose the same random number. Fortunately, if the number of sources in a session that need to choose random numbers is small compared to the square root (8192) of the size of the space (2**26), the probability is "negligibly small" according to the analysis of the Birthday Problem.
- If hosts on a private internet reuse IP addresses that are assigned to hosts in the Internet, and if the two internets are connected by some kind of translator (e.g., a firewall), then the applications running in the private internet would have to be configured to use random identifiers when communicating across the translator. This could be messy.

There are two mechanisms that might be used for conflict resolution:

- During the relatively long startup period when a participant first joins a conference, the IDs of the other participants are likely to be heard before the new participant transmits, so there is a chance for the new participant to choose another random number if a conflict is heard.
- If a new site begins using an ID in conflict with an existing one, then any site in the session can send a message challenging the new participant (since the owner of the ID might not be in the session at that moment). Randomized delays would be used to prevent an implosion of responses. This challenge mechanism would need to be specified in the protocol.

If the probability of unresolved conflict is smaller than the probability of lightning striking your workstation, it probably doesn't matter.

This scheme would impose a limit on session size, but the practical limit for the "lightweight session" mechanism is somewhere between 1K and 10K anyway. Beyond that, the scaling of control packet rate backoff stops working (it becomes slower than about 2 minutes, and this gets unstable if participants come and go at similar time scales). For larger sessions, such as cable TV distribution, some means of aggregation would have to be specified, and that mechanism could provide a partitioning of the identifier space as well.

For some applications, such as wb, the identifiers should be stable across invocations to avoid loss of ownership of previously generated information. If a random ID must be chosen then it must be remembered in persistent storage (e.g., a file). Rules such as this for the use of identifiers might be part of an application-specific profile.

Multiple sources participating in the same session on one host would need some means to coordinate which one gets to use the IP address and which ones must select random identifiers. Although vat does not do this yet, I think Van has a plan.

Advantages

- Loop detection.
- Less work in translators and bridges.
- May be required for some encryption or authentication schemes under the new option mechanism.

Disadvantages

- 32 bits instead of 16.
- Must be convinced by probability.

7. Control packet format / Reception reports

With control and data being carried on separate ports, the functions of the RTCP options would be moved into the control packets. The primary functions are:

- Providing information about the sender, e.g., name
- Providing reception reports for all sources received
- Relating the sender's media timestamp to real time, and also marking the time of the reception reports

In previous AVT meetings, it has been suggested that RTCP might be removed from the RTP spec entirely. However, Van Jacobson states the position that in order for RTP to be used on a large scale, we must provide mechanisms for network service providers as well as users to evaluate the distribution quality, and the mechanism is to monitor the reception reports generated from the multicast data itself as the test traffic. This mechanism needs to be considered a fundamental part of RTP having to do with using the network for distribution; it's use should not be considered optional. Since the reception report mechanism is independent of particular media, it goes in the base RTP spec rather than a profile. The spec should be written such that any application using RTP will work in multicast mode, with unicast as a special case.

The format of control packets has not been as well defined as the other items in this proposed collection of changes. The header format for the control packets need not be exactly the same as for the data packets, but it may be useful to keep them similar. The following information is proposed:

```
    Sender info:
        media timestamp
        NTP timestamp
        sender's packet count
        sender's byte count
        sender's name
    Reception info:
        number of reception reports
        array of report structures:
            source identifier
            count of packets received
            count of bytes received
            variance of interarrival time
            last timestamp received from this source in a session packet
            delay time since that session packet was received
```

The reception report structure includes multiple reports per packet (all sources heard from since the last report). If the conference is large such that the control packet rate is slowed down, and if there are so many sources generating traffic that the reports will not all fit into a packet, then a different set of sites would be picked each time in round robin order, for as many as will fit.

The last two items in the reception report can be used in an NTP-like algorithm to figure the round-trip propagation delay and then divide by two. Then one can make an ordering of the sources based on propagation delay as an approximation of distance, and can cluster the sites based on error rate and on distance.

The reception report should contain absolute information rather than deltas so it does not matter if a report is missed. The NTP timestamp in the sender information is useful for media synchronization, but it is also needed as the timestamp at which the counts in the reception report were generated. This double duty requires that the media timestamp have sufficient resolution that one can be generated to correspond to the NTP timestamp at the time a reception report is generated; alternatively, the reception report must be generated at an instant that corresponds to a media timestamp tick. Given two reception reports with timing information allows the counts to be translated into rates. The RTP QOS option includes minimum and maximum delay measures. These are not included above because an outlyer can make the value useless as a description of the distribution.

The general guideline for the construction of the control packet is to put more common information first so that application-independent monitors can process all the common information without having to know anything about the format of the application- or media-dependent information. So, the packet would be assembled as:

```
    sender info:
        common
        application dependent
        media dependent

    reception report:
        common, for all sources received
        application dependent, for all sources received
        media dependent, for all sources received
```

There is one field that tells the number of reception reports, and then the reception info is contained in several parallel arrays, all with the same number of entries (some of which may be structs) and all indexed by the same number. The application dependent part would be defined by a profile and the main spec wouldn't say what the format was.

In addition to the sender description and reception report information, the existing RTCP defines the following options. These options may need to be incorporated into the control packet structure in some way:

FMT (format description) -- Allows format codes to be defined dynamically. This may also be accomplished with a higher-layer session protocol, but some applications may not include such a protocol.

BYE (goodbye) -- Indicates that a source is terminating its participation in a session. Since no source description or reception report information is required, this could be a separate (trivial) control packet format.

APP (application-specific controls) -- Application-specific sections of the sender information and reception reports in the control packet provide a means to carry application-specific information defined by a profile. However, in cases where multiple implementations of a single application interoperate but may have their own control information to communicate, and additional option mechanism may be appropriate.

[end]

IETF-Audio Video Transport WG Minutes

```
From rem-conf-request@es.net Mon May  2 01:53:42 1994
Posted-Date: Sun 1 May 94 23:54:41 PDT
Date: Sun 1 May 94 23:54:41 PDT
From: Stephen Casner
Subject: AVT Working Group IETF29 Seattle minutes
To: rem-conf@es.net
Mail-System-Version:
Content-Length: 36215
```

Reported by Steve Casner/USC-ISI

Minutes of Audio/Video Transport Working Group (AVT)

1. Overview

The meeting began with a brief report on the status of the **Real-time Transport Protocol** *(RTP)*. The draft RTP specification that was submitted with a request for "IESG Last Call" just before the previous IETF meeting in November. The review by the Transport Area Directorate called for several changes so that RTP would more closely follow the principles of Application Level Framing (ALF) and Integrated Layer Processing (ILP) proposed by David Clark and David Tennenhouse in their SIGCOMM '90 paper. In two discussions among Steve Casner, Ron Frederick and Van Jacobson in which the vat and nv programs were taken as design examples, the following list of proposed changes was constructed:

- Carry the control and data traffic on separate ports
- Remove the application-level multiplexing of the Channel ID and move it to an encapsulation for the cases where it's needed
- Minimize the use of options
- Make the definition of some fields application-specific (in particular, the timestamp clock rate and sync marker)
- Use global rather than local IDs, to be able to detect loops
- Specify more precisely how reception reports should be provided

These changes were described in more detail by Steve Casner in a long message sent to the rem-conf@es.net mailing list on 21-Mar-94. The first 90 minutes of the first session and the beginning of the second session were occupied by a presentation of these changes. The sections below interleave the main points from the presentation of proposed changes with issues from the discussion that followed. The working group generally agreed with the changes, although some remaining details were identified. A summary appears at the end of these minutes.

Our task is now to complete the design to address these details, update the specification and get consensus from the working group via email. Steve Casner will take responsibility for sending out a more complete draft of the proposed changes to start the discussion. Everyone is encouraged to participate. Please send email if there were issues not addressed, to the rem-conf list or to casner@isi.edu. The goal is to submit the draft for Last Call after review at the July

IETF meeting in Toronto.

Part of the Directorate response after reviewing the RTP submission in November was a recommendation that the protocol be given Experimental status rather than Proposed Standard. However, John Wroclawski said that was only a strategy for dealing with the differences of opinion about the protocol, and that with the proposed changes he believes the Directorate would not have a preference. John and others expressed personal preferences for Proposed Standard status as the goal, once the details of the changes are completed and integrated into the specification. No objections were voiced.

At the end of the second session, there were two additional presentations. Julio Escobar gave a report on the use of RTP to support the Synchronization Protocol developed at BBN. The only requirement not satisfied by RTP was the need to communicate the calculated synchronization delays among the destination processors; this could be added to the control packet as application-specific data.

Don Hoffman gave a brief description of the two Internet Drafts just released on encoding profiles for Cell-B and MPEG/MPEG-2. Cell-B was developed by Sun but is freely available. MPEG-2 is in development as an ISO/IEC standard. In the MPEG profile, two formats are proposed, one for interoperation with other transport mechanisms using the MPEG-2 Systems environment, and a second, simpler format intended for native Internet-style applications. Comments on the drafts were solicited. The drafts are:

    draft-speer-cellb-rtp-encap-00.txt
    draft-hoffman-rtp-mpeg-encap-00.txt

These encoding profiles will will be finalized after the RTP changes are resolved, and reviewed at the Toronto IETF meeting.

There has also been further work on the JPEG encapsulation since the November meeting. It is now proposed that a small header be included at the start of the data area in each packet. This would be a more compact representation of some of the information in a JFIF header. With this header, decoding restarts would be possible, allowing partial frames to be decoded. Those who are interested in more details should send email to Bill Fenner (fenner@cmf.nrl.navy.mil).

2. Proposed changes to RTP

The sections below interleave the main points from the presentation of proposed changes with issues from the discussion that followed. However, much of the presentation of advantages and disadvantages has been omitted here; those who did not read the message describing the changes are advised to fetch it from ftp.isi.edu, directory /mbone/avt, file rtpv2-proposal.txt. In the same directory is rtpv2-presentation.ps which contains the slides from this meeting.

2.1. Control and data on separate ports

This change removes RTCP functions from RTP data packets and puts them into a separate packet stream on another port to streamline data processing and to allow monitoring programs to receive the control information with having to sort through the data.

No issues regarding this point were brought up in the discussion. However, in email Jarmo Molsa expressed the concern that using separate ports would require an additional socket on many systems, and that on some systems the number of sockets is limited.

2.2. Remove Channel ID, put multiplexing into encapsulations as needed

There has been quite a bit of debate about the Channel ID in the past, but none during this meeting. Most applications that may have used multiple channel IDs could use multiple destination ports instead.

2.3. Replace data options with a bit field

With the RTCP operations separated into another packet stream on a different port, there are only a few optional fields to be selected in data packets. That allows the optional fields to be indicated by bits in the fixed header. If the bits are also grouped together, it is possible to interpret them as a "packet format", which is a single demultiplexing point for processing as a set of fixed formats if appropriate. The packet format method might not be used in software implementations except perhaps for a fast path when all options are off; however, for hardware, the bit-field approach would be much simpler than the option structure as defined in the current RTP spec.

The RTP options would be replaced as follows:

CSRC (content source) -- A bit field of 5 or 6 bits in the fixed header to count the number of content source identifiers to follow, for packets produced by a bridge (mixer). Zero indicates that the packet did not come from a bridge. This count takes the place of the length field in the current CSRC option.

SSRC (synchronization source) -- It was decided that the SSRC identifier should always be included (see the discussion of global identifiers below), so this option becomes part of the fixed header.

BOS (beginning of sync unit) -- Eliminated as an option.

APP (application-specific) -- Application specific functions could be defined in profiles as extensions to the RTP header, but there is no provision for one implementation of an application to define its own optional information in a way that other implementations can simply ignore.

SDST (sync destination, or reverse-path option) -- Reverse path packets would be eliminated because translators would not keep a table of source transport address to SSRC identifier mappings, and the SSRC/SDST would not be visible in encrypted packets.

ENC (encryption) -- When encryption is used, the whole RTP unit is encrypted (all of the RTP header and data). The receiver depends upon header validity checks (version, format, sequence number, and timestamp having reasonable values) to discard packets that should have been decrypted (or decrypted with a different key). There is no provision for an explicit initialization vector; instead zero would be used with random initial values for the sequence number and timestamp to deter a known plaintext attack. The key and encryption algorithm would be specified by out of band information; key switching can be done by decrypting just the header with the old key first, and if the validity checks fail, then trying again with the new key.

MIC (authentication) -- An authenticated packet would be indicated by a bit in the header which would indicate the presence of an authentication field later in the header. With the global identifier scheme, the SSRC could be authenticated to have been set by one of the sources (but could still be false). The authentication method (covered by ENC, keyed, symmetrically encrypted, or asymmetrically encrypted), algorithm and key (if any) would be known from out-of-band information. As with encryption, key changes could be accomplished by trying with old an new keys in succession. Alternatively, a key descriptor could be included at the start of the authentication field. To allow some receivers to ignore the authentication without knowing the out-of-band information, a length field would be needed at the start of the authentication field.

Some details for the encryption and authentication methods have not been fully specified. Encryption requires an indication of how much padding was added to the end of the data to round up the length to a multiple of the encryption block size. Christian Huitema also suggested that the authentication digest be appended as a trailer rather than a header encapsulation to allow for hardware processing or to allow a single loop through the data for decryption, authentication, and decompression, per the ILP design principle. It may be possible to use one bit in the header to indicate whether there is any trailer present, and then to structure the trailer to indicate padding and authentication digest. In email, Mark Wahl brought up the desire to include a full NTP timestamp and sender description (e.g., distinguished name) within the authentication to minimize replay and support non-repudiation. These details are to be designed and presented later.

The other fields in the first 32 bits of the RTP header are the version number, format, sync marker, and sequence number. It is proposed that the version number be bumped to 2 if these proposed changes are adopted. The format field would remain unchanged. The sync marker definition would change (see below), but would remain one bit. The sequence number should stay at 16 bits for arithmetic convenience, but could be trimmed if necessary. So, the resulting header bit count would be:

```
   6  for CSRC count
   1  for encryption/authentication trailer
   2  for version
   6  for format
   1  for sync marker
  16  for sequence number
 ---
  32
```

Jon Crowcroft said the UCL monitoring software uses sequence numbers to measure loss and reordering, so reducing the sequence number space to 8 bits would cause a problem for their sampling mechanism (over a long view, not looking at all packets). Van Jacobson commented that misordering by more than the sequence number space can still be corrected using the timestamp. The purpose of the sequence number is to detect gaps.

Jon also requested that space be made for one additional bit to indicate the congestion probe options used in Ian Wakeman's congestion control scheme. Several others expressed concern in offline comments that there should be some mechanism for adding functions or options to the data packets. This is an issue that needs further discussion.

2.4. Media-specific timestamps

To eliminate the requirement for RTP applications to know the time of day and some possible but rare off-by-1 errors in timestamp tick rate conversion, it is proposed that the rate at which the clock ticks, instead of always being 65536Hz, would become a parameter of the format. For some formats, such as the variable frame rate video we are now using, it may make sense to retain the 65536Hz rate, while for most audio formats, the clock rate would be set to be the same as the sampling rate, as is done for the timestamp in vat.

It is also proposed to remove the requirement for senders to make timestamp adjustments. For example, the timestamps would just follow the input device's sample counter. For senders that do know the time of day, control packets would carry both an RTP timestamp (sample clock) and the corresponding full 64-bit NTP timestamp to establish the relationship.

Bill Fink noted that with the previous definition of the RTP timestamp as the middle 32 bits of an NTP timestamp, it would be simple for a recorder to replay an RTP stream with the original timing or to synchronize the playback of two RTP streams without any knowledge of the encodings. He asked if recorders would now have to be media specific.

There are several possibilities. A recorder can replay packets with the same timing as they were received simply by recording its own local timestamps along with the packets. However, that requires extra space, and does not remove the jitter induced on the path from the source to the recorder. The recorder can reconstruct the source's timing in several ways: 1) from the media timestamps and the nominal rate if it is known; 2) by calculating the nominal rate based on the difference between times in a pair of control packets relating media timestamps to real time; or 3) by calculating the nominal rate based on the arrival rate when the packets are received.

2.5. Application-specific sync marker bit

In the current RTP specification, the sync bit is defined to mark the last packet of a synchronization unit. The proposed change is to allow the meaning of the sync marker bit to be defined by an application profile. The application profile could further specify that the bit is defined by each of the encoding profiles to be used under that application profile.

In email, Henning Schulzrinne suggested that two sync marker bits be defined in the fixed header, one for start and one for end of synchronization unit, since both events are interesting, as opposed to making the one bit variable and requiring the application to define what it means. However, the general sentiment in the meeting was in favor of one flexible bit as proposed because there may be more than these two meanings, e.g. in video there may be a difference between sync unit and frame. The recommendation was to remove the word "sync" from the description of the bit, making it an application-specific marker to be defined as seen fit in the application profile.

2.6. Global source identifiers

RTP currently uses locally unique source 16-bit identifiers to keep track of distinct sources when packets flow through translators and bridges and lose their original transport addresses. The identifiers are locally assigned by the translators and bridges and are remapped by successive translators or bridges. The proposed change is based on the scheme used by vat and wb in which sources assign 32-bit identifiers that are globally unique within a particular medium in a particular session. No remapping is required; loops can be detected.

The way the proposed scheme would be specified is that the identifiers are 32-bit opaque numbers, and that there may be a variety of means to assign them, including choosing them randomly. Since there may be collisions in a distributed allocation mechanism, there is a collision resolution protocol, but the allocation mechanism should be such that the collision resolution protocol is typically exercised less than 1% of the time. Two mechanisms are proposed:

- When a participant first joins a conference, the identifiers of the other participants are likely to be heard before the new participant transmits, so there is a chance to choose another random number if there's a conflict.

- If a new site begins using an identifier in conflict with an existing one, then any site in the session can send a message challenging the new participant (since the owner of the identifier might not be in the session at that moment). Randomized delays would be used to prevent an implosion of responses. This challenge mechanism would need to be specified in the protocol.

2.6.1 Use of IPv4 addresses

The topic receiving the most discussion during the meeting was Van Jacobson's proposal that in the current Internet multicast environment, one way to assign unique identifiers is to use the IPv4 address of the host if there is only one source per host. Van argued that this is almost always the case because sessions tend to be associated with a particular set of hardware resources, for example, the mike and speaker associated with a particular workstation. Even if multiple X terminals that have audio hardware operate off a single workstation, the X terminal also has an IP address which can be used.

For those case where there are additional sources on one host, a identifier would be chosen at random from the "class F" IP address space (26 bits) so as not to conflict with assigned addresses. Christian Huitema pointed out that we could use the IP multicast class D space (28 bits) since a multicast address should never be a source address. However, a bit of additional analysis done since the meeting by Sally Floyd in response to a question from Steve Casner shows that it may be better just to choose the random numbers from the 32-bit space anyway. For a session with 2000 participants, the probability of collision is lower in the separate 26-bit space only if the number of random allocations is less than 60, and in that case the probability of collision is less than 0.00005 either way.

Using the IPv4 address for most identifiers has two advantages:

1) The identifier field could be omitted and a single bit could be used in the RTP header to denote that the identifier was implicit in the common case that the identifier was the same as the source address of the packet (a significant reduction for audio).

2) Given that IP addresses are (generally) unique, this reduces the probability of identifier collisions by requiring fewer random allocations (compared to allocating all identifiers randomly).

Since RTP may also be used over protocols other than IP, Jon Crowcroft suggested that the spec allow the use of unique addresses from any address space, for example, the bottom 32 bits of an IEEE 802 address. Unfortunately, this idea runs into trouble when we consider interoperation among multiple networks with different address spaces. Ron Frederick and Abel Weinrib pointed out that two spaces may have allocation patterns such that the rate of collision is much higher than would be the result of allocating all identifiers randomly. For example, consider a session involving multiple enterprises connected through firewalls and each using the small portion of the IPv4 space allocated to private networks as described in RFC 1597. Even when the session size is small, some individuals who communicate frequently might always collide (because the addresses are fixed).

Therefore, only one address space can be allowed to us use non-random assignment unless the spaces can be arranged to be non-overlapping, and that seems impractical. It seems technically bad to give the optimization to IPv4 if we expect this protocol to live for a while, not to mention the political concern that anything that says "IPv4 address" in it is probably short-sighted.

Ron also disagreed with Van's contention that the presence of multiple sources on one host would necessarily be infrequent. For example, one might run multiple copies of nv for multiple cameras or for a camera and an X screen capture source. Many applications would require the extra mechanism to support figuring out if the IPv4 address could be used or if a random identifier must be allocated.

The conclusion was that the specification should not suggest the use of IPv4 addresses or any other fixed mapping of an address space to the 32-bit identifier space. Instead, applications should always allocate identifiers randomly in the full 32-bit space. A combination of the address and time might be used to seed the random number generator.

Given that the synchronization source (SSRC) identifier is to be chosen randomly, that identifier must always be included explicitly in the RTP header. While many of us hate to see the header length increased from 8 to 12 bytes, there was a consensus that this was the right decision because it simplifies several other aspects of the protocol: it means we don't need a flag bit to indicate inclusion of the SSRC field; it allows translators to forward encrypted and/or authenticated packets without any special handling; and it eliminates any dependence upon the packet source address (e.g., IP address).

2.6.2 Permanence of identifiers

One of the advantages that may have been cited for using the IPv4 address as the identifier is its permanence. When an application using random identifiers is killed and restarted in the same session, a person may be identified as a new source unless the application stored the previously allocated identifier somewhere. However, it is also wrong to assume that the IPv4 address would serve as a persistent identifier for a participant. Van Jacobson pointed out that there may be multiple people using one machine serially in one session, and they should be identified uniquely; John Wroclawski noted that the IP address may change between application invocations due to mobile computing and dynamically assigned addresses. The specification must state that it is mandatory for an application that cares about reuse to provide a way to save the identifier itself, independent of what scheme was used to select the identifier in the first place.

Ron Frederick said that even if IPv4 addresses were used as identifiers, some other source might might allocate the same number randomly or through some other means. Therefore, applications must always be prepared to go through the collision resolution protocol. Abel Weinrib took the point further, saying that assumptions of permanence are dangerous. If an application saves an identifier and tries to reuse it sometime later, another source may have begun using that identifier in the meantime so the reuse would be seen as a collision and the first application would be stuck.

For applications that require long-term permanence or for which information ownership has serious consequences, a higher-level participant name, such as may be carried in the RTP control packet or a higher-level control protocol, should be the point of ownership and the RTP source identifiers should be considered transient. Bill Fink noted that the use of these identifiers is very dependent upon how the control protocol works and how the session and directory information is maintained.

2.6.3 Scaling limit for random identifier allocation

The primary disadvantage of random identifier allocation is the possibility that two sources will choose the same number; this is the same as the Birthday Problem. Steve Casner presented two graphs of probability of collision based on analysis and bits of Mathematica supplied by Sally Floyd. For identifiers picked uniformly from a 32-bit space, the probability of there being at least one collision is 0.0001 for 1K identifiers chosen or 0.01 (1%) for 10K identifiers chosen. Considering that not all the identifiers in a session are chosen at once, it is useful to consider that the probability of collision is much smaller for adding one participant to a session in which the identifiers are already known not to conflict.

So, for most applications the maximum session size for which the random allocation scheme performs acceptably would be somewhere in the range 1K-10K. This is not a new constraint since it matches the limit on session size already imposed by the rate backoff limit for the sender identification and reception report packets. However, it is important to insure that the collision resolution protocol scales or fails gracefully when the expected limit is exceeded.

Some sessions, such as IETF meetings and Space Shuttle missions, are already near 1K in the cumulative list of participants, though the maximum number of simultaneous participants is around 200-300. We are likely to reach the limit within the expected life of the protocol. To go beyond that will require some additional mechanism such as aggregation and hierarchy. Van Jacobson suggested that in a large, private broadcast system, one might assign customer identifiers as people sign up, and use those as globally unique identifiers. Or, in truly unidirectional systems, the receivers may not require unique identifiers because they cannot emit any packets.

John Wroclawski believes there is a clear architectural understanding of how to make this scale to much larger systems, and there is some engineering work that's beginning to get underway. This may well be an appropriate future direction for this protocol and this working group, but we don't know the general solution yet. Perhaps the spec should include an applicability statement for the random identifier scheme as defined, along with some hints about the architectural ideas and work in progress, to encourage experimentation.

2.7. Control packet format / Reception reports

With control and data being carried on separate ports, the functions of the RTCP options would be moved into the control packets. The primary functions are:

- Providing information about the sender, e.g., name

- Providing reception reports for all sources received since the last report, or round robin through the list for as many as will fit; reports should contain absolute information rather than deltas so it does not matter if a report is missed.

- Relating the sender's media timestamp to real time for intermedia synchronization, also marking the time of the reception reports so rates can be calculated from differences in packet and byte counts

The format of control packets has not been as well defined as the other items in the proposed collection of changes. The header format for the control packets need not be exactly the same as for the data packets, but it may be useful to keep them similar. SSRC, CSRC, encryption and authentication are required for control packets as well as data packets, and the mechanisms should be the same. The general guideline for the construction of the control packet is to put more common information first so that application-independent monitors can process all the common information without having to know anything about the format of the application- or media-dependent information. The length of any application-dependent section would be implicit in the profile that defines the application. However, for those applications that may include media-dependent information, an identifier and possibly a length field for the particular media format would be required.

2.7.1 Mandatory reception reports

In previous AVT meetings, it has been suggested that RTCP might be removed from the RTP spec entirely. However, Van Jacobson argues that in order for RTP to be used on a large scale in the Internet, we must provide mechanisms for network service providers as well as users to evaluate the distribution quality. The multicast data itself is the test traffic, and the reception reports are the monitoring mechanism. This mechanism needs to be considered a fundamental part of RTP covering all applications; it's use should not be considered optional. The RTP spec and application profiles should be written such that any application using RTP will work in multicast mode, with unicast as a special case. In an IP multicast environment, reception reports must be sent by all receivers, including those that never send any data. The rate at which the reports are generated is scaled back as the number of participants increases so that the aggregate control traffic does not exceed the bandwidth of one data stream. The control packets would be sent too slowly to be useful beyond a session size somewhere in the 1K-10K range.

This brought up the question of using RTP in truly unidirectional systems in which it is presumed that successful transmission is assured by some means and in which it may not be possible for receivers to send a report. In such a system, reception reports would not be required. The specification will need some applicability statements for those pieces that may not be architecturally relevant as we move into new applications. A profile may specify how to use RTP in a unidirectional environment.

2.7.2 Requirement for NTP time in control packet

Frank Kastenholz expressed concern about the requirement for an NTP timestamp in the control packet. This needs to be clarified: systems that do not know what time it is should not be required to put an absolute time into that field. However, if the calculation of the data rate and packet rate from the counts in the reception report depends upon the time field, then there may be a requirement for a time value that increments at close to real time even if it is not related to the time of day. An audio application might be able to keep track of time by referencing the sample counter, but what about applications with aperiodic data?

2.7.3 Congestion control

Jon Crowcroft and Christian Huitema would like to enable receivers to more quickly report that they are experiencing loss. Jon sought support for Ian Wakeman's congestion control scheme which selects receivers to report more frequently than the rate backoff of reception reports would allow in a large session. Christian wanted the reception report to include an explicit measurement of packet loss by the receiver so that when loss is observed in small sessions the sender can be advised to adapt. Van Jacobson claims that transients are not a concern, and that one cannot robustly make a better diagnosis than the proposed reception report mechanism allows without making the report size much larger. He offered to provide convincing arguments offline. We are awaiting results of those discussions.

2.7.4 Reports for multiple-priority streams

John Wroclawski asked whether the reception reports would provide an accurate estimate of overall loss rate for multi-priority schemes and layered encodings. One may only care about the highest priority packets, or one may want more information than the general reception report provides. Additional information could be added in the media-dependent section of the report, but this might not solve the problem for generic monitoring tools. Van Jacobson said that the different priority data streams should be sent on different multicast addresses so the loss rates can be reported separately. However, monitors would still need to know that the loss rate on lower priority streams is expected to be higher. This question needs more thought.

2.7.5 Additional control functions

The sender description and reception report information correspond to the RTCP SDES and QOS options. In addition, the functions of the FMT, BYE and APP options are to be retained, they need to be incorporated into the control packet structure in some way. These functions could be specified by different packet types in the control packet header, plus another type for the mandatory sender description / reception report. Greg Minshall suggested that there be some mechanism for tagging along additional information on the report packets that must be sent periodically anyway. One way would be to allow multiple control packets to be aggregated into one lower-layer packet.

FMT (format description) -- Allows format codes, which go in the 6-bit field in the data packet header, to be defined dynamically in addition to the small set that are predefined. FMT is not intended for redefinition of codes to allow switching among more than 32 formats in one session; applications needing to do that should define a format whose data field begins with a further specification of the format. FMT may be seldom used since a higher-layer session protocol, such as a session directory, would normally be used to select the formats for a session and define any format codes not already predefined.

BYE (goodbye) -- Indicates that a source is terminating its participation in a session. Since no source description or reception report information is required, this could be a separate (trivial) control packet format. Greg Minshall suggested that we might want the BYE packet to include the reason for leaving.

APP (application-specific controls) -- Application-specific sections of the sender information and reception reports in the control packet already provide a means to carry periodic information defined by an application profile. However, multiple implementations of a single application profile may interoperate but still need to communicate information that may be ignored by other implementations. Each profile could define on its own a mechanism to include implementation-dependent information, but it seems better to have a common mechanism. Implementation-specific data could be added at end of the reception reports after the media-dependent data. To avoid conflicts among multiple implementations each of which wanted to include its own information, some implementation identification such as the ASCII name in the existing APP option is required. A separate APP packet type would use the same implementation identification mechanism for transporting signals that are not periodic and have nothing to do with the sender information.

Charley Kline asked whether we should simply require that applications use some more sophisticated session control protocol if they need to communicate any control information beyond what's in the sender description and reception reports. For many applications, it is likely to be more convenient to use the RTCP path. The basic question is, shall we make a provision for RTP to operate in the absence of any higher-layer mechanism for some applications?

3. Summary of agreements and open issues

The following items were agreed during the meeting:

- Global identifiers would always be 32-bit random numbers; the IPv4 address should not be used.
- The sync source (SSRC) identifier would always be included in a new field added to the fixed RTP header.
- Encryption should cover whole packet; zero would be used for the initialization vector, with random initial values for timestamp and sequence number.
- The above agreements imply elimination of reverse-path packets.
- The word "sync" would be dropped from the name of the marker bit; its meaning would be defined by the application profile.
- The plan is to seek Proposed Standard status for the protocol.

The following open issues were identified during the discussion:

- Need to define how to indicate the length of padding required for encryption and the format of a trailer for authentication.
- Should an optional congestion probe field be defined?
- Should there be a means to add options to data packet?
- The control packet format needs to be defined in detail.
- Specify how timestamps are used on systems that don't know time.
- Should a receiver-calculated loss measure be in reception reports?
- What is the effect of multi-priority streams on loss monitoring?
- Should the FMT and APP options be included? (should RTP be usable without a higher-layer session protocol?)
- An applicability statement for RTP needs to be defined.

Internet Engineering Task Force
INTERNET-DRAFT
draft-ietf-avt-rtp-04.txt

Audio-Video Transport WG
H. Schulzrinne/S. Casner
AT&T/ISI

October 20, 1993
Expires: 12/31/93

RTP: A Transport Protocol for Real-Time Applications

Status of this Memo

See also draft-ietf-avt-issues-*.{ps,txt}

Abstract

This memorandum describes the real-time transport protocol, RTP. RTP provides end-to-end network transport functions suitable for applications transmitting real-time data, such as audio, video or simulation data over multicast or unicast network services. RTP does not address resource reservation and does not guarantee quality-of-service for real-time services. The data transport is augmented by a control protocol (RTCP) designed to provide minimal control and identification functionality particularly in multicast networks. Within multicast associations, sites can also direct control messages to individual sites. RTP and RTCP are designed to be independent of the underlying transport and network layers. The protocol supports the use of RTP-level translators and bridges.

This specification is a product of the Audio/Video Transport working group within the Internet Engineering Task Force. Comments are solicited and should be addressed to the working group's mailing list at rem-conf@es.net and/or the authors.

1 Introduction

This memorandum specifies the real-time transport protocol (RTP), which provides end-to-end delivery services for data with real-time characteristics, for example, interactive audio and video. RTP itself does not provide any mechanism to ensure timely delivery or provide other quality-of-service guarantees, but relies on lower-layer services to do so.

It does not guarantee delivery or prevent out-of-order delivery, nor does it assume that the underlying network is reliable and delivers packets in sequence. The sequence numbers included in RTP allow the end system to reconstruct the sender's packet sequence, but sequence numbers might also be used to determine the proper location of a packet, for example in video decoding, without necessarily decoding packets in sequence. RTP is designed to run on top of a variety of network and transport protocols, for example, IP, ST-II or UDP.(footnote available) RTP transfers data in a single direction, possibly to multiple destinations if supported by the underlying network. A mechanism for sending control data in the opposite direction, reversing the path traversed by regular data, is provided.

While RTP is primarily designed to satisfy the needs of multi-participant multimedia conferences, it is not limited to that particular application. Storage of continuous data, interactive distributed simulation, active badge, and control and measurement applications may also find RTP applicable. Profiles are used to instantiate certain header fields and options for particular sets of applications (see Section 9). A profile for audio and video data may be found in the companion Internet draft draft-ietf-avt-profile.

This document defines a packet format shared by two protocols:

- the real-time transport protocol (RTP), for exchanging data that has real-time properties. The RTP header consists of a fixed-length portion plus optional control fields;
- the RTP control protocol (RTCP), for conveying information about the participants in an on-going session. RTCP consists of additional header options that may be ignored without affecting the ability to receive data correctly. RTCP is used for "loosely controlled" sessions, i.e., where there is no explicit membership control and set-up. Its functionality may be subsumed by a session control protocol, which is beyond the scope of this document.

A discussion of real-time services and algorithms for their implementation and background on some of the RTP design decisions can be found in the current version of the companion Internet draft draft-ietf-avt-issues.

The current Internet does not support the widespread use of real-time services. High-bandwidth services using RTP, such as video, can potentially seriously degrade other network services. Thus, implementors should take appropriate precautions to limit accidental bandwidth usage. Application documentation should clearly outline the limitations and possible operational impact of high-bandwidth real-time services on the Internet and other network services.

2 RTP Use Scenarios

The following sections describe some aspects of the use of RTP. The examples were chosen to illustrate the basic operation of applications using RTP, not to limit what RTP may be used for. In these examples, RTP is carried on top of IP and UDP, and follows the conventions established by the profile for audio and video specified in the companion Internet draft draft-ietf-avt-profile.

2.1 Simple Multicast Audio Conference

A working group of the IETF meets to discuss the latest protocol draft, using the IP multicast services of the Internet for voice communications. Through some allocation mechanism, the working group chair obtains a multicast group address; all participants use the destination UDP port specified by the profile. The multicast address and port are distributed, say, by electronic mail, to all intended participants. The mechanisms for discovering available multicast addresses and distributing the information to participants are beyond the scope of RTP.

The audio conferencing application used by each conference participant sends audio data in small chunks of, say, 20 ms duration. Each chunk of audio data is preceded by an RTP header; RTP header and data are in turn contained in a UDP packet. The Internet, like other packet networks, occasionally loses and reorders packets and delays them by variable amounts of time. To cope with these impairments, the RTP header contains timing information and a sequence number that allow the receivers to reconstruct the timing seen by the source, so that, in our case, a chunk of audio is delivered to the speaker every 20 ms. The sequence number can also be used by the receiver to estimate how many packets are being lost. Each RTP packet also indicates what type of audio encoding (such as PCM, ADPCM or GSM) is being used, so that senders can change the encoding during a conference, for example, to accommodate a new participant that is connected through a low-bandwidth link.

Since members of the working group join and leave during the conference, it is useful to know who is participating at any moment. For that purpose, each instance of the audio application in the conference periodically multicasts the name, email address and other information of its user. Such control information is carried as RTCP SDES options within RTP messages, with or without audio data (see Section 6.2). These periodic messages also provide some indication as to whether the network connection is still functioning. A site sends the RTCP BYE (Section 6.3) option when it leaves a conference. The RTCP QOS (Section 6.4) option indicates how well the current speaker is being received and may be used to control adaptive encodings.

2.2 Bridges

So far, we have assumed that all sites want to receive audio data in the same format. However, this may not always be appropriate. Consider the case where participants in one area are connected through a low-speed link to the majority of the conference participants, who enjoy high-speed network access. Instead of forcing everyone to use a lower-bandwidth, reduced-quality audio encoding, a bridge is placed near the low-bandwidth area. This bridge resynchronizes incoming audio packets to reconstruct the constant 20 ms spacing generated by the sender, mixes these reconstructed audio streams, translates the audio encoding to a lower-bandwidth one and forwards the lower-bandwidth packet stream to the low-bandwidth sites.

After the mixing, the identity of the high-speed site that is speaking can no longer be determined from the network origin of the packet. Therefore, the bridge inserts a CSRC option (Section 5.2.1) into the packet containing a list of short, locally unique site identifiers to indicate which site(s) contributed to that mixed packet. An example of this is shown for bridge B1 in Fig. 1. As name and location information is received by the bridge in SDES options from the high-speed sites, that information is passed on to the receivers along with a mapping to the CSRC identifiers. This also works if an RTP packet is mixed through several bridges, with the CSRC value being mapped into a new locally unique value at each bridge. For example, in Fig. 1 bridge B3 maps CSRC value 3 for packets coming from B2 into CSRC value 1 for packets going to T2.

2.3 Translators

Not all sites are directly accessible through IP multicast. For these sites, mixing may not necessary, but a translation of the underlying transport protocol is. RTP-level gateways that do not restore timing or mix packets from different sources are called translators in this document. Application-level firewalls, for example, will not let any IP packets pass. Two translators are installed, one on either side of the firewall, the outside one funneling all multicast packets received through the secure connection to the translator inside the firewall. The translator inside the firewall sends them again as multicast packets to a multicast group restricted to the site's internal network. Other examples include the connection of a group of hosts speaking only IP/UDP to a group of hosts that understand only ST-II.

After RTP packets have passed through a translator, they all carry the network source address of the translator, making it impossible for the receiver to distinguish packets from different speakers based on network

```
       [E1]                                      [E6]
         |                                         |
  E1:17  |                               E6:15     |
         |                                         |   E6:63/6
         V   B1:48 (1,2)           B1:28/1 (1,2)   V   B1:63/5 (1,2)
        (B1)------------->----------------->---------------->[E7]
         ^                 ^       E4:28/2         ^   E4:63/3
   E2:1  |         E4:47   |                       |   B3:63/4 (1,4)
         |                 |                       |
        [E2]              [E4]     B3:89 (1,4)     |
                                                   |      LEGEND:
 [E3]   --------->(B2)----------->(B3)-------------|    [End system]
          E3:64        B2:12 (3)    ^                   (Bridge)
                                    | E5:45
                                    |
                                   [E5]        source: port/SSRC (CSRCs)
                                               ------------------------>
```

Figure 1: Sample RTP network with end systems, bridges and translators

source addresses. Since each sending site has its own sequence number space and slightly offset timestamp space, the receiver could not properly mix the audio packets. (For video, it could not properly separate them into distinct displays.) Instead of forcing all senders to include some globally unique identifier in each packet, a translator inserts an SSRC option (Section 5.2.2) with a short identifier for the source that is locally unique to the translator. This also works if an RTP packet has to travel through several translators, with the SSRC value being mapped into a new locally unique value at each translator. An example is shown in Fig. 1, where hosts T1 and T2 are translators. The RTP packets from host E4 are identified with SSRC value 2, while those coming from bridge B1 are labeled with SSRC value 1. Similarly, translator T2 has labeled packets from E6, B1, E4 and B3 with SSRC values 6, 5, 3 and 4, respectively.

2.4 Security

Conference participants would often like to ensure that nobody else can listen to their deliberations. Encryption, indicated by the presence of the ENC option (Section 5.5.1), provides that privacy. The encryption method and key can be changed during the conference by indexing into a table. For example, a meeting may go into executive session, protected by a different encryption key accessible only to a subset of the meeting participants.

For authentication, a number of methods are provided, depending on needs and computational capabilities. All these message integrity check (MIC) options (Sections 5.5.3 and following) compute cryptographic checksums, also known as message digests, over the RTP data.

3 Definitions

Payload is the data following the RTP fixed header and any RTP/RTCP options. The payload format and interpretation are beyond the scope of this memo. RTP packets without payload are valid. Examples of payload include audio samples and video data.

An RTP packet consists of the encapsulation specific to a particular underlying protocol, the fixed RTP header, RTP and RTCP options, if any, and the payload, if any. A single packet of the underlying protocol may contain several RTP packets if permitted by the encapsulation method.

A (protocol) port is the "abstraction that transport protocols use to distinguish among multiple destinations within a given host computer. TCP/IP protocols identify ports using small positive integers." [1] The transport selectors (TSEL) used by the OSI transport layer are equivalent to ports.

A transport address denotes the combination of network address, e.g., the 4-octet IP Version 4 address, and the transport protocol port, e.g., the UDP port. In OSI systems, the transport address is called transport service access point or TSAP. The destination transport address may be a unicast or multicast address.

A content source is the actual source of the data carried in an RTP packet, for example, the application that originally generated some audio data. Data from one or more content sources may be combined into a single RTP packet by a bridge, which becomes the synchronization source (see next paragraph). Content source identifiers carried in CSRC options identify the logical source of the data, for example, to highlight the current speaker in an audio conference; they have no effect on the delivery or playout timing of the data itself. In Fig. 1, E1 and E2 are the content sources of the data received by E7 from bridge B1, while B1 is the synchronization source.

A synchronization source may be a single content source, or the combination of one or more content sources, produced by a bridge, with its own timing. Each synchronization source has its own sequence number space. The audio coming from a single microphone and the video from a camera are examples of synchronization sources. The receiver groups packets by synchronization source for playback. Typically a single synchronization source emits a single medium (e.g., audio or video). A synchronization source may change its data format, e.g., audio encoding, over time. Synchronization sources are identified by their transport address and the identifier carried in the SSRC option. If the SSRC option is absent, a value of zero is assumed for that identifier.

A transport source is the transport-level origin of the RTP packets as seen by the receiving end system. In Fig. 1, host T2, port 63 is the transport source of all packets received by end system E7.

A channel comprises all synchronization sources sending to the same destination transport address using the same RTP channel ID.

An end system generates the content to be sent in RTP packets and consumes the content of received RTP. An end system can act as one or more synchronization sources. (Most end systems are expected to be a single synchronization source.) When a packet is transmitted from an end system, the end system is the content source, synchronization source, and transport source at that point.

An (RTP-level) bridge receives RTP packets from one or more sources, combines them in some manner and then forwards a new RTP packet. A bridge may change the data format. Since the timing among multiple input sources will not generally be synchronized, the bridge will make timing adjustments among the streams and generate its own timing for the combined stream. Therefore, when a packet is processed through a bridge, the bridge becomes the synchronization source as well as the transport source, but the originating end system remains the content source for that data. As the bridge combines packets from multiple content sources into a single outgoing packet, each of the contributing content sources is noted by the insertion of an identifier into the CSRC option in the outgoing packet. Audio bridges and media converters are examples of bridges. In Fig. 1, end systems E1 and E2 use the services of bridge B1. B1 inserts CSRC identifiers for E1 and E2 when they are active (e.g., talking in an audio conference). The RTP-level bridges described in this document are unrelated to the data link-layer bridges found in local area networks. If there is possibility for confusion, the term 'RTP-level bridge' should be used. The name bridge follows common telecommunication industry usage.

An (RTP-level) translator forwards RTP packets, but does not alter their sequence numbers or timestamps. Examples of its use include encoding conversion without mixing or retiming, conversion from multicast to unicast, and application-level filters in firewalls. A translator is neither a synchronization nor a content source, but does become the transport source for packets which flow through it. The properties of bridges and translators are summarized in Table 1. Checkmarks in parentheses designate possible, but unlikely actions. The RTP options are explained in Section 5.2, the RTCP options in Section 6.

A synchronization unit consists of one or more packets that are emitted contiguously by the sender. The most common synchronization units are talkspurts for voice and frames for video transmission. During playout synchronization, the receiver must reconstruct exactly the time difference between packets within a synchronization unit (in the case of video, all the packets of a frame are typically given the same timestamp so there is no time difference). The time difference between synchronization units may be changed by the receiver to compensate for clock drift or to adjust to

	end sys.	bridge	translator
mix sources	--	x	--
change encoding	--	x	x
encrypt	x	x	(x)
sign for authentication	x	x	--
alter content	x	x	x
insert CSRC (RTP)	--	x	--
insert SSRC (RTP)	(x)	(x)	x
insert SDST (RTP)	x	x	--
insert SDES (RTCP)	x	x	--

Table 1: The properties of end systems, bridges and translators

changing network delay jitter. For example, if audio packets are generated at fixed intervals during talkspurts, the receiver tries to play back packets with exactly the same spacing. However, if, for example, a silence period between synchronization units (talkspurts) lasts 600 ms, the receiver may adjust it to, say, 500 ms without this being noticed by the listener.

Non-RTP mechanisms refers to other protocols and mechanisms that may be needed to provide a useable service. In particular, for multimedia conferences, a conference control application may distribute multicast addresses and keys for encryption and authentication, negotiate the encryption algorithm to be used, and determine the mapping from the RTP format field to the actual data format used. For simple applications, electronic mail or a conference database may also be used. The specification of such mechanisms is outside the scope of this memorandum.

4 Byte Order, Alignment, and Reserved Values

All integer fields are carried in network byte order, that is, most significant byte (octet) first. This byte order is commonly known as big-endian. The transmission order is described in detail in [2], Appendix A. Unless otherwise noted, numeric constants are in decimal (base 10). Numeric constants prefixed by `0x' are in hexadecimal.

Fields within the fixed header and within options are aligned to the natural length of the field, i.e., 16-bit words are aligned on even addresses, 32-bit long words are aligned at addresses divisible by four, etc. Octets designated as padding have the value zero. Fields designated as "reserved" or R are set aside for future use; they should be set to zero by senders and ignored by receivers.

Textual information is encoded accorded to the UTF-2 encoding of the ISO standard 10646 (Annex F) [3,4]. US-ASCII is a subset of this encoding and requires no additional encoding. The presence of multi-octet encodings is indicated by setting the most significant bit to a value of one. An octet with a binary value of zero may be used as a string terminator for padding purposes. However, strings are not required to be zero terminated.

5 RTP Data Transfer Protocol

5.1 RTP Fixed Header Fields

The RTP header has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Ver| ChannelID |P|S|  format    |       sequence number         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     timestamp (seconds)       |      timestamp (fraction)     |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
| options ...                                                   |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

The first eight octets are present in every RTP packet and have the following meaning:

protocol version: 2 bits Identifies the protocol version. The version number of the protocol defined in this memo is one (1).

channel ID: 6 bits The channel identifier field forms part of the tuple identifying a channel (see definition in Section 3) to provide an additional level of multiplexing at the RTP layer. The channel ID field is convenient if several different channels are to receive the same treatment by the underlying layers or if a profile allows for the concatenation of several RTP packets on different channels into a single packet of the underlying protocol layer (see Section 8.1).

option present bit (P): 1 bit This flag has a value of one (1) if the fixed RTP header is followed by one or more options and a value of zero (0) otherwise.

end-of-synchronization-unit (S): 1 bit This flag has a value of one in the last packet of a synchronization unit, a value of zero otherwise. As shown in Appendix A, the beginning of a synchronization unit can be readily established from

.this flag.(footnote available)

format: 6 bits The format field forms an index into a table defined through the RTCP FMT option or non-RTP mechanisms (see Section 3). The mapping establishes the format of the RTP payload and determines its interpretation by the application. Formats defined through the FMT option must be kept in a separate mapping table per sender as there can be no guarantee that all senders will use the same table. If no mapping has been defined through these mechanisms, a standard mapping is specified by the profile in use by the application in question. An initial set of default mappings for audio and video is specified in the companion profile document RFC TBD, and may be extended in future editions of the Assigned Numbers RFC.

sequence number: 16 bits The sequence number counts RTP packets. The sequence number increments by one for each packet sent. The sequence number may be used by the receiver to detect packet loss, to restore packet sequence and to identify packets to the application.

timestamp: 32 bits The timestamp reflects the wall clock time when the RTP packet was generated. Several consecutive RTP packets may have equal timestamps if they are generated at once. The timestamp consists of the middle 32 bits of a 64-bit NTP timestamp, as defined in RFC 1305 [5]. That is, it counts time since 0 hours UTC, January 1, 1900, with a resolution of 65536 ticks per second. (UTC is Coordinated Universal Time, approximately equal to the historical Greenwich Mean Time.) The RTP timestamp wraps around approximately every 18 hours.

The timestamp of the first packet within a synchronization unit is expected to closely reflect the actual sampling instant, measured by the local system clock. If possible, the local system clock should be controlled by a time synchronization protocol such as NTP. However, it is allowable to operate without synchronized time on those systems where it is not available, unless a profile or session protocol requires otherwise. It is not necessary to reference the local system clock to obtain the timestamp for the beginning of every synchronization unit, but the local clock should be referenced frequently enough so that clock drift between the synchronized system clock and the sampling clock can be compensated for gradually. Within one synchronization unit, it may be appropriate to compute timestamps based on the logical timing relationships between the packets. For audio samples, for example, it is more accurate to maintain the time within a synchronization unit in samples, incrementing by the number of samples per packet, and then converting to an RTP timestamp (see Appendix A.1).

5.2 The RTP Options

The RTP fixed packet header may be followed by options and then the payload. Each option consists of the F (final) bit, the option type designation, a one-octet length field denoting the total number of 32-bit words comprising the option (including F bit, type and length), followed by any option-specific data. The last option before the payload has the F bit set to one; for all other options this bit has a value of zero.

An application may discard options with types unknown to it. The option type number range is divided into four regions. Types 0 through 31 are general RTP options, their syntax and semantics independent of the format or any profile. Types 32 through 63 are RTCP options, again independent of format and profile. Types 64 through 95 are specific to a particular profile, i.e., valid for a range of formats. Types 96 through 126 are specific to a format as defined by the four-character name registered with the Internet Assigned Numbers Authority (IANA); these options may be described in a profile or format specification. Format-specific options are parsed according to the format selected by the format field in the fixed RTP header, as shown in Appendix A.4. Types 0 through 126 are reserved and registered with IANA. Type 127 is defined in Section 5.3 of this document; it allows application-specific extensions not registered with IANA.

Unless otherwise noted, each option may appear only once per packet. Each packet may contain any number of options. Options may appear in any order, unless specifically restricted by the option description. In particular, the position of some security options has significance.

5.2.1 CSRC: Content source identifiers

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|F|  CSRC = 0    |     length     | content source identifiers  ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The CSRC option, inserted only by bridges, lists all sources that contributed to the packet. For example, for audio packets, all sources that were mixed together to create a packet are enumerated, allowing correct talker indication at the receiver. The CSRC option may contain one or more 16-bit content source identifiers. The identifier values must be unique for all content sources received from a particular synchronization source on a particular channel; the value of binary zero is reserved and may not be used. If the number of content sources is even, the two octets needed to pad the list to a multiple of four octets are set to zero. There should be no more than one CSRC option within a packet. If no CSRC option is present, the content source identifier is assumed to have a value of zero. CSRC options are not modified by translators. If a bridge receives a packet containing a CSRC option from another bridge located upstream, the identifier values in that CSRC option must be translated into new, locally unique values.

A conformant RTP implementation does not have to be able to generate or interpret the CSRC option.

5.2.2 SSRC: Synchronization source identifier

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|F|  SSRC = 1    |  length = 1   |          identifier           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The SSRC option may be inserted by translators, end systems and bridges. It is typically used only by translators, but it may be used by an end system application to distinguish several sources sent with the same transport source address. If packets from multiple synchronization sources will be transmitted with the same transport source address (e.g., the same IP address and UDP port), an SSRC option must be inserted in each packet with a distinct identifier for the synchronization source. Conversely, synchronization sources that are distinguishable by their transport address do not require the use of SSRC options. The SSRC value zero is reserved and would not normally be transmitted; if received, the SSRC option should be treated as if not present. When no SSRC option is present, the transport source address is assumed to indicate the synchronization source. There must be no more than one SSRC option per packet; thus, a translator must remap the SSRC identifier of an incoming packet into a new, locally unique SSRC identifier. The SSRC option can be viewed as an extension of the source port number in protocols like UDP, ST-II or TCP.

An RTP receiver must support the SSRC option. RTP senders only need to support this option if they intend to send more than one source to the same channel using the same source port. For example, a translator could use multiple source ports rather than insert SSRC options, but this is likely to be less convenient.

5.2.3 BOS: Beginning of synchronization unit

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|F|   BOS = 3    |   length = 1  |        sequence number        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The sequence number contained within this option is that of the first packet within the current synchronization unit. The BOS option allows the receiver to compute the offset of a packet with respect to the beginning of the synchronization unit, even if the last packet of the previous synchronization unit was lost. It is expected that many applications will be able to tolerate such a loss, and so will not use the BOS option but rely on the S bit. Those applications which do require the BOS option may use a profile that specifies it is always included.

5.3 APP: Application-specific option

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|F|  APP = 127  |    length     |           subtype             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          name (ASCII)                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   application-dependent data                ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

subtype: 2 octets May be used as a subtype to allow a set of APP options to be defined under one unique name, or for any application-dependendent data.

name: 4 octets A name chosen by the person defining the set of APP options to be unique with respect to other APP options this application might receive. The application creator might choose to use the application name, and then coordinate the allocation of subtype values to others who want to define new options for the application. Alternatively, it is recommended that others choose a name based on the entity they represent, then coordinate the use of the name within that entity. The name is interpreted as a sequence of four ASCII characters, with uppercase and lowercase characters treated as distinct.

application-dependent data: variable length Application-dependent data may or may not appear in an APP option. It is interpreted by the application and not RTP itself.

The APP option is intended for experimental use as new applications and new features are developed, without requiring option type value registration. APP options with unrecognized names should be ignored. After testing and if wider use is justified, it is recommended that each APP option be redefined without the subtype and name fields and registered with the Internet Assigned Numbers Authority using an option type in the RTP, RTCP, profile-specific, or format-specific range as appropriate.

5.4 Reverse-Path Option

With two-party (unicast) communications, having a receiver of data relay back control information to the sender is straightforward. Similarly, for multicast communications, control information can easily be sent to all members of the group. It may, however, be desirable to send a unicast message to a single member of a multicast group, for example to send a reception quality report. For such purposes, RTP includes a mechanism for sending so-called reverse RTP packets. The format of reverse RTP packets is exactly the same as for regular RTP packets and they can make use of any option defined in this memorandum, except SSRC, as appropriate. The support for and semantics of particular options are to be specified by a profile or an individual application. Additional options may be defined as prescribed in Section 5.2 as needed for a particular profile, format or application.

Reverse RTP packets travel through the same translators as forward RTP packets. When RTP is carried in a protocol that provides transport-level addressing (ports), a site may distinguish reverse RTP packets from forward RTP packets by their arrival port. Reverse RTP packets arrive on the same port that the site uses as a source port for forward (data) RTP packets. Therefore, that port should be assigned uniquely; in particular, it should be different than the destination port used with the multicast address, and if the application is participating in several multicast groups, a distinct source port should be used to send to each group. If RTP is carried directly within IP or some other network-layer protocol that does not include port numbers, the reverse RTP packet must include an SDST option (defined next), and the presence of the SDST option signals that the packet is a reverse RTP packet.

A receiver of reverse RTP packets cannot rely on sequence numbers being consecutive, as a sender is allowed to use the same sequence number space while communicating through this reverse path with several sites. In particular, a receiver of reverse RTP packets cannot tell by the sequence numbers whether it has received all reverse RTP packets sent to it. As a consequence, it is expected that reverse RTP packets would carry only options and no payload. The sequence number space of reverse RTP packets has to be completely separate from that used for RTP packets sent to the multicast group. If the same sequence number space were used, the members of the multicast group not receiving reverse RTP packets would detect a gap in their received sequence number space. The sender of reverse RTP packets should ensure that sequence numbers are unique, modulo wrap-around, so that they can, if necessary, be used for matching request and response. (Currently, no such request-response mechanism has been defined.) As a hypothetical example, consider defining a request to pan the remote video camera. After completing the request, the receiver of the request would send a generic acknowledgement containing the sequence number of the request back to the requestor as an option (not as the packet sequence number in the fixed header).

The timestamp should reflect the approximate sending time of the packet. The channel ID must be the same as that used in the corresponding forward RTP packets.

If many receivers send a reverse RTP packet in response to a stimulus in the data stream, for example a request for retransmission of a particular data frame, the simultaneous delivery of a large number of packets back to the data source can cause congestion for both the network and the destination (this is known as an "ack implosion"). Thus reverse RTP packets should be used with care, perhaps with mechanisms such as response rate limiting and random delays to spread out the simultaneous delivery.

5.4.1 SDST: Synchronization destination identifier

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|F|  SDST = 2   |  length = 1   |          identifier           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The SDST option is only inserted by RTP end systems and bridges if they want to send a unicast packet to a particular site within the multicast group. These are called reverse RTP packets. Only reverse RTP packets may include the SDST option, but not all reverse RTP packets require it, as explained below. A reverse RTP packet must not contain an SSRC option.

If a forward RTP packet carries SSRC identifier X when sent from A to B, where A and B may be either two translators or an end system and a translator, the unicast reverse RTP packet will carry an SDST option with identifier X from B to A.

Consider the topology shown in Fig. 1. Assume that RTP is carried over a network or transport protocol that includes port numbers and that all forward RTP packets are addressed to destination port 8000. For the case that B1 wants to send a reverse packet to E1, B1 simply sends to the source address and port, that is, port 17 in this example. E1 can tell by the arrival on port 17 that the packet is a reverse packet rather than a regular (forward) packet. No SDST option is required. The mechanism is somewhat more complicated when translators intervene. We focus on end system E7. E7 receives, say, video from a range of sources, E1 through E6 as indicated by the arrows. The transmission from T2 to E7 could be either multicast or unicast. Assume that E7 wants to send a retransmission request, a request to pan the camera, etc., to end system E4 alone. E7 may not be able to directly reach E4, as E4 may be using a network protocol unknown to E7 or be located behind a firewall. According to the figure, video transmissions from E4 reach E7 through T2 with source port 63 and SSRC identifier 3. For the reverse message, E7 sends a message to T2, with destination port 63 and SDST identifier 3. T2 can look up in its table that it sends forward data coming from T1 with that SSRC identifier 3. T2 also knows that those messages from T1 carry SSRC 2 and arrive with source port 28. Just like E7, T2 places the SSRC identifier, 2 in this case, into the SDST option and forwards the packet to T1 at port 28. Finally, translator T1 consults its table to find that it labels packets coming from E4, port 47 with SSRC value 2 and thus knows to forward the reverse packet to E4, port 47. T1 can either place value zero into the SDST option or remove the option. Note that E4 cannot directly determine that E7 sent the reverse packet, rather than, say, E6. If that is important, a global identifier as defined for the QOS option (Section 6.4) needs to be included in some option in the reverse packet.

When reverse RTP packets are carried directly within IP or some other network-layer protocol that does not include port numbers, the SDST option is required to distinguish reverse RTP packets from forward RTP packets. In the case where no SSRC identifier needs to be placed in the SDST option, the value zero should be inserted.

Only applications that need to send or receive reverse control RTP packets need to implement the SDST option.

5.5 Security Options

The security options below offer message integrity, authentication and confidentiality and the combination of the three. Support for the security options is not mandatory, but the encryption option (ENC) should at least be recognized to avoid processing encrypted data. The four message integrity check options --- MIC, MICA, MICK and MICS --- are mutually exclusive, i.e., only one of them should be used in a single RTP packet. Multiple options are provided to satisfy varying security requirements and computational capabilities.

A variety of security services may be provided with the encryption option, one of the message integrity check options, or the combination of the two options:

confidentiality: Confidentiality means that only the intended receiver(s) can decode the received RTP packets; for others, the RTP packet contains no useful information. Confidentiality of the content is achieved by encryption. The presence of encryption and the encryption initialization vector is indicated by the ENC option.(footnote available)

authentication and message integrity: These two security services are provided by the message integrity check options. The receiver can ascertain that the claimed originator is indeed the originator of the data (authentication) and that the data within an RTP packet has not been altered after leaving the sender (message integrity). Through examination of the timestamp and sequence number fields in the RTP header to verify that all the packets of a sequence are present and played in order, an implementation may also establish the integrity of that packet sequence.

The services offered by MICA and MIC/MICK/MICS differ: With MIC/MICK/MICS, the receiver can only verify that the message originated within the group holding the secret key, rather than authenticate the sender of the message. The MICA option, in combination with certificates(footnote available), affords true authentication of the sender. The certificates for MICA must be distributed through means outside of RTP.

authentication, message integrity, and confidentiality: By carrying both the message integrity check and ENC option in RTP packets, the authenticity, message integrity and confidentiality of the packet can be assured (subject to the limitations discussed in the previous paragraph).

For this combination of security features when group authentication is sufficient, the combination ENC and MIC is recommended (instead of MICS or MICK), as it yields the lowest processing overhead.

All message integrity check options carry a message digest, which is a cryptographic hash function that transforms a message of any length to a fixed-length byte string, where the fixed-length string has the property that it is computationally infeasible to generate another, different message with the same digest. The message digest is computed over the fixed header, a portion of the message integrity check option itself (the first two octets for MICA, the first four octets for MIC and MICS, or the whole option in the case of MICK), and the remaining header options and payload that will immediately follow the message integrity check option in the RTP packet. The fixed header is protected to foil replay attacks and reassignment to a different channel.

When both a message integrity check option and the ENC option are to be included, the recommended ordering is that the message integrity check be applied first as described in the previous paragraph. Then the message integrity check option and the remaining header options and payload that follow it are encrypted using the shared secret key. The ENC option is prepended to the encrypted data; that is, the ENC option must be followed immediately by the message integrity check option, without any other options in between. The receiver first decrypts the octets following the ENC option and then authenticates the decrypted data using the message digest contained in the message integrity check option.

For the MIC option in particular, this ordering must be used because the ENC option is required to provide confidentiality of the message digest. For the other message integrity check options, this ordering allows explicit detection of an encryption key mismatch. However, both the decryption and message integrity check functions must be performed before an invalid packet can be detected, which increases the potential for a denial-of-service attack. For those applications where this is a concern, the ordering may be reversed.

The message integrity check options and the ENC option must not cover the SSRC or SDST option, i.e., SSRC or SDST must be inserted between the fixed header and the ENC or message integrity check option; SSRC and SDST are subject to change by translators that likely do not possess the necessary descriptor table (see below) and encryption keys. Trusted translators that have the necessary keys and descriptor translation table may modify the contents of the RTP packet, unless the MICA option is used (see MICA description in Section 5.5.3).

All security options except MICA carry a one-octet descriptor field. These descriptors are indexes into two tables, one for the message integrity check options and one for the ENC option, established by non-RTP means to contain the digest algorithms (MD2, MD5, etc.), encryption algorithms (DES variants) and encryption keys or shared secrets (for the MICK option) to be used during a session. The descriptor value may change during a session, for example, to switch to a different encryption key. The tables must be established to be the same for all sources within the same channel; this reduces per-site state information.

The descriptor value zero selects a set of default algorithms, namely, MD5 for the message digest algorithm and DES in CBC mode for encryption algorithm, so that basic security features may be implemented using simple non-RTP mechanisms to communicate a single shared secret (key). For example, the key might be communicated by telephone or (private) email and entered manually.

5.5.1 ENC: Encryption

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|F|   ENC = 8   |   length = 3  |    reserved   |   descriptor  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     DES (CBC) initialization vector, bytes 0 through 3        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     DES (CBC) initialization vector, bytes 4 through 7        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|F|   ENC = 8   |   length = 1  |    reserved   |   descriptor  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Every encrypted RTP packet must contain this option in one of the two forms shown. All octets in the packet following this option are encrypted, using the encryption key and symmetric encryption algorithm selected by the descriptor field. Note that the fixed header is specifically not encrypted because some fields must be interpreted by translators that will not have access to the key. The descriptor value may change over time to accommodate varying security requirements or limit the amount of ciphertext using the same key. For example, in a job interview conducted across a network, the candidate and interviewers could share one key, with a second key set aside for the interviewers only. For symmetric keys, source-specific keys offer no advantage.

The descriptor value zero is reserved for a default mode using the Data Encryption Standard (DES) algorithm in CBC (cipher block chaining) mode, as described in Section 1.1 of RFC 1423 [7]. The padding specified in that section is to be used. In the first form of the ENC option, the 8-octet initialization vector (IV) is carried unencrypted within the option, but must be generated uniquely for each packet. In the second form (indicated by an option length of one), the ENC option does not contain an initialization vector and instead the fixed RTP header is used as the initialization vector. (Using the fixed RTP header as the initialization vector avoids regenerating the initialization vector for each packet and incurs less header overhead; it is unique for a period of at least 18 hours.) For details on the tradeoffs for CBC initialization vector use, see [8]. Support for encryption is not required. Implementations that do not support encryption should recognize the ENC option so that they can avoid processing encrypted messages and provide a meaningful failure indication. Implementations that support encryption should, at the minimum, always support the

-DES algorithm in CBC mode.

5.5.2 MIC: Messsage integrity check

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|F|   MIC = 9   |    length     |   reserved    |  descriptor   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 message digest (unencrypted)                 ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The MIC option option is used only in combination with the ENC option immediately preceding it to provide confidentiality, message integrity and group membership authentication. The message integrity check uses the digest algorithm selected by the descriptor field. The value zero implies the use of the MD5 message digest. Note that the MIC option is not separately encrypted.

5.5.3 MICA: Message integrity check, asymmetric encryption

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|F|  MICA = 10  |    length     |          message digest      ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    (asymmetrically encrypted)                ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The message digest is asymmetrically encrypted using the sender's private key according to the algorithm defined for Privacy Enhanced Mail, described in Section 4.2.1 of RFC 1423 (here "MIC" denotes the general term "message integrity check", not the RTP option):

As described in PKCS #1, all quantities input as data values to the RSAEncryption process shall be properly justified and padded to the length of the modulus prior to the encryption process. In general, an RSAEncryption input value is formed by concatenating a leading NULL octet, a block type BT, a padding string PS, a NULL octet, and the data quantity D, that is, RSA input value = 0x00, BT, PS, 0x00, D. To prepare a MIC for RSAEncryption, the PKCS #1 "block type 01" encryption-block formatting scheme is employed. The block type BT is a single octet containing the value 0x01 and the padding string PS is one or more octets (enough octets to make the length of the complete RSA input value equal to the length of the modulus) each containing the value 0xFF. The data quantity D is comprised of the MIC and the MIC algorithm identifier which are ASN.1 encoded.

For the purposes of the MICA option, the encoding of data quantity D may be considered as a fixed binary sequence identifying the message integrity check algorithm, followed by the octets of the message digest. Currently, only the use of the MD2 and MD5 algorithms is defined, as described in RFC 1319 [9] (as corrected in Section 2.1 of RFC 1423) and RFC 1321 [10], respectively. For MD2, the fixed binary sequence (shown here in hexadecimal) is 0x30, 0x20, 0x30, 0x0C, 0x06, 0x08, 0x2A, 0x86, 0x48, 0x86, 0xF7, 0x0D, 0x02, 0x02, 0x05, 0x00, 0x04, 0x10, and for MD5 it is 0x30, 0x20, 0x30, 0x0C, 0x06, 0x08, 0x2A, 0x86, 0x48, 0x86, 0xF7, 0x0D, 0x02, 0x05, 0x05, 0x00, 0x04, 0x10. The appropriate sequence is followed immediately by the message digest, which is 16 octets long for both MD2 and MD5. For clarification of the octet ordering of the message digest, see RFC 1423, Sections 2.1 and 2.2.

As an example, for an RSA encryption modulus length of 512 bits or 64 octets, the RSA input value would be:

```
           BT  <---- PS --->      <------------ D ------------->
      0x00 0x01 0xFF ... 0xFF 0x00 0x30 ... 0x10 [message digest]
                (29 octets)        (18 octets)     (16 octets)
```

The length of the encrypted message digest will be equal to the modulus of the RSA encryption used, rounded to the next integral octet count. Contrary to what is specified in RFC 1423 for Privacy Enhanced Mail, the asymmetrically encrypted message digest is carried in binary, not represented in the printable encoding of RFC 1421, Section 4.3.2.4. The encrypted message digest is inserted into the MICA option immediately following the length octet, and is padded at the end to make the MICA option a multiple of four octets long. The value of the padding is left unspecified. The number of non-padding bits within the signature is known to the receiver as being equal to the key length.

The modulus and public key are conveyed to the receivers by non-RTP means. After the message digest is decrypted, the message integrity check algorithm is identified through the octets prepended to the actual 16-octet digest.

Asymmetric keys are used since symmetric keys would not allow authentication of the individual source in the multicast case. A translator is not allowed to modify the parts of an RTP packet covered by the MICA option as the receiver would have no way of establishing the identity of the translator and thus could not verify the integrity of the RTP packet.

5.5.4 MICK: Message integrity check, keyed

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|F|  MICK = 11  |     length    |    reserved   |   descriptor  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           message digest (including shared secret)          ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

This message integrity check option does not require encryption, but includes a shared secret in the computation of the message digest. The shared secret is equivalent to the key used for the MICS and ENC options, but is 16 octets long, padded if needed with binary zeroes. The shared secret is first placed into the MICK option where the message will later go, then the digest is computed over the fixed RTP header, the whole MICK option including the shared secret, and the remaining header options and payload that will immediately follow the MICK option in the RTP packet. The shared secret in the MICK option is then replaced by the computed 16-octet message digest for transmission.

The receiver stores the message digest contained in the MICK option, replaces it with the shared secret key and computes the message digest in the same manner as the sender. If the RTP packet has not been tampered with and has originated with one of the holders of the shared secret, the computed message digest will agree with the digest found on reception in the MICK option.(footnote available)

The digest algorithm and shared secret are selected by the descriptor field. The value zero implies the use of the MD5 message digest and a single shared secret.

5.5.5 MICS: Message integrity check, symmetric-key encrypted

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|F|  MICS = 12  |     length    |    reserved   |   descriptor  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           message digest (symmetrically encrypted)          ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

This message integrity check encrypts the message digest using the DES algorithm in ECB mode as described in RFC 1423, Section 3.1. The digest algorithm and symmetric key are selected by the descriptor field. The value zero implies the use of the MD5 message digest and a single key.

6 RTP Control Protocol --- RTCP

The RTP control protocol (RTCP) conveys minimal control and advisory information during a session. It provides support for "loosely controlled" sessions, i.e., where participants enter and leave without membership control and parameter negotiation. The services provided by RTCP augment RTP, but an end system does not have to implement RTCP features to participate in sessions. There is one exception to this rule: if an application sends FMT options, the receiver has to decode these in order to properly interpret the RTP payload. RTCP does not aim to provide the services of a session control protocol and does not provide some of the services desirable for two-party conversations. If a session control protocol is in use, the services of RTCP should not be required. (As of the writing of this document, a session or conference control protocol has not been specified within the Internet.)

RTCP options share the same structure and numbering space as RTP options, which are described in Section 5. Unless otherwise noted, control information is carried periodically as options within RTP packets, with or without payload. RTCP packets are sent to all members of a session, typically using multicast. These packets are part of the same sequence number space as RTP packets not containing RTCP options. The period should be varied randomly to avoid synchronization of all sources and its mean should increase with the number of participants in the session to limit the growth of the overall network and host interrupt load. The length of the period determines, for example, how long a receiver joining a session has to wait until it can identify the source. A receiver may remove from its list of active sites a site that it has not been heard from for a given time-out period; the time-out period may depend on the number of sites or the observed average interarrival time of RTCP messages. Note that not every periodic message has to contain all RTCP options; for example, the EMAIL part within the SDES option might only be sent every few messages. RTCP options should also be sent when information carried in RTCP options changes, but the generation of RTCP options should be rate-limited.

6.1 FMT: Format description

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|F|  FMT = 32   |    length     |R|R|  format   |    reserved   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          format name                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    format-dependent data                    ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

format: 6 bits The format field corresponds to the index value from the format field in the RTP fixed header, with values ranging from 0 to 63.

format name: 4 octets The format name describes the format in an unambiguous way and is registered with the Internet Assigned Numbers Authority. The format name is interpreted as a sequence of four ASCII characters, with uppercase and lowercase characters treated as distinct. Format names beginning with the letter 'X' are reserved for experimental use and not subject to registration.

format-dependent data: variable length Format-dependent data may or may not appear in a FMT option. It is interpreted by the application and not RTP itself.

A FMT mapping changes the interpretation of a given format value carried in the fixed RTP header starting at the packet containing the FMT option. The new interpretation applies only to packets from the same synchronization source as the packet containing the FMT option. If format mappings are changed through the FMT option, the option should be sent periodically as otherwise sites that did not receive the FMT option due to packet loss or joining the session after the FMT option was sent will not know how to interpret the particular format value.

6.2 SDES: Source descriptor

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|F|  SDES = 34  |    length     |       source identifier       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| type=PORT (2) |   length = 1  |             port              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| type=ADDR (1) |    length     |   reserved    | address type  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     network-layer address                   ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| type=PORT (2) |   length > 1  |   reserved    |   reserved    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             port                            ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| type=CNAME (4)|    length     | user and domain name        ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| type=EMAIL (5)|    length     | electronic mail address     ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| type=NAME (6) |    length     | common name of source       ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| type=LOC (8)  |    length     | geographic location of site ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| type=TXT (16) |    length     | text describing source      ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| type=PRIV(255)|   length > 1  |            subtype            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          name (ASCII)                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   application-defined content               ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The SDES option is composed of the first four octets shown concatenated with one or more of the subsequent items as described individually below. SDES provides a mapping between a numeric source identifier and those items, which describe a particular source.(footnote available) For those applications where the size of a multi-item SDES option would be a concern, multiple SDES options may be formed with subsets of the items to be sent in separate packets.

When an SDES option originates from a content source (the actual source of the data), the identifier value is zero. If the data flows through a bridge, the bridge forwards the SDES information, but changes the SDES source identifier to the value used in the CSRC option when identifying that content source. The bridge may choose to store the SDES information received from a content source and change then number of items sent together or the rate at which SDES information is sent. A bridge uses an identifier value of zero within an SDES option to describe itself rather than the content sources bridged by it, but if a bridge contributes local data to outgoing packets, it should select another non-zero source identifier for that traffic and send CSRC and SDES options for it as well.

Translators do not modify or insert SDES options. The end system performs the same mapping it uses to identify the content sources (that is, the combination of transport source, SSRC identifier and the source identifier within this

. SDES option) to identify a particular source. SDES information is specific to traffic from a source on a particular channel, unless a profile or a higher-layer control protocol defines that the same SDES describes traffic from that source on some set of channels.

Each item has a structure similar to that of RTP and RTCP options, that is, a type field followed by a length field, measured in multiples of four octets. No final bit (see Section 5.2) is needed since the overall length is known. Item types 0 through 127 apply to all profiles, while types 128 through 254 are allocated to profile-specific items; both ranges are reserved and registered with the Internet Assigned Numbers Authority (IANA). Item type 255 (PRIV) is provided for private or experimental extensions not registered with IANA. Items are independent of the format value.

All of the SDES items are optional and unrecognized items may be ignored; however, if quality-of-service monitoring is to be used, receivers will require the PORT and ADDR items from the SDES option in order to construct the QOS option. Only the TXT item is expected to change during the duration of a session. Text items are encoded according to the rules in Section 4. Items are padded with the binary value zero to the next multiple of four octets. Each item may appear only once unless otherwise noted. A description of the content of these items follows:

PORT/ADDR: The PORT item contains the source transport selector, such as the UDP source port number, and the ADDR item contains the network address of the source, for example, the IP version 4 address or an NSAP. Both are carried in binary form, not as "dotted decimal" or similar human-readable form. Address types are identified by the Domain Name Service Resource Record (RR) type, as specified in the current edition of the Assigned Numbers RFC.

There must be no more than one PORT item in an SDES option. The PORT item should be followed immediately by an ADDR item. Concatenated, these two items serve as a globally unique identifier for the source which is returned in the QOS option. As far as RTP is concerned, this identifier is opaque, so it is unimportant which address is used for multi-homed hosts. Applications may find the address or port useful for debugging or monitoring, but should not assume that the combination can be used to communicate with the source process because it may be on the other side of a firewall or using a different transport protocol. If it is useful to the application, it is permissible for a source to include additional ADDR items after the first to convey additional addresses if the source is multi-homed, or if the source's address may be represented in multiple schemes, for example during the transition from IPv4 to IPng.(footnote available)

The figure shows the PORT item in two forms. The first form shows the concatenated PORT and ADDR items as they would be used for the TCP and UDP protocols. For an IPv4 address, the length of the ADDR item would be 2, and the address type would be 1. The second form of the PORT item is indicated by a length field greater than one and is used when the transport selector (port number) is larger than two octets. Octets three and four of the item are reserved (zero) and the transport selector appears in words two and following of this item, in network byte order.

CNAME: Canonical user and host identifier, e.g.,

```
        "doe@sleepy.megacorp.com" or "sleepy.megacorp.com".
```

The CNAME item must have the format "user@host" or "host", where "host" is the fully qualified domain name of the host from which the real-time data originates, formatted according to the rules specified in RFC 1034, RFC 1035 and Section 2.1 of RFC 1123. The "host" form may be used if a user name is not available, for example on single-user systems. The user name should be in a form that a program such as "finger" or "talk" could use, i.e., it typically is the login name rather than the real-life name. Note that the host name is not necessarily identical to the electronic mail address of the participant.

EMAIL: User's electronic mail address, formatted according to RFC 822, for example,

```
        "John:Doe@megacorp.com".
```

NAME: Common name describing the source, e.g., "John Doe, Bit Recycler, Megacorp". This name may be in any form desired by the user. For applications such as conferencing, this form of name may be the most desirable for display in participant lists, and therefore might be sent most frequently (profiles may establish such priorities).

LOC: Geographic user location. Depending on the application, different degrees of detail are appropriate for this item. For conference applications, a string like "Murray Hill, New Jersey" may be sufficient, while, for an active badge system, strings like "Room 2A244, AT&T BL MH" might be appropriate. The degree of detail is left to the implementation and/or user, but format and content may be prescribed by a profile.

TXT: Text describing the source, e.g., "out for lunch".

PRIV: Private, unregistered items. The subtype and name fields are to be used in the same manner as in the APP option (Section 5.3). The format and content of the octets following the name field are determined by the application defining the item.

6.3 BYE: Goodbye

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|F|   BYE = 35   | length = 1    |   content source identifier   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The BYE option indicates that a particular session participant is no longer active. When a BYE option originates from a content source (the actual source of the data), the identifier value is zero. If the message flows through a bridge, the bridge forwards the BYE message, but changes the identifier to be the (non-zero) value used in the CSRC option when identifying that content source. If a bridge shuts down, it should first send BYE options for all content sources it handles, followed by a BYE option with an identifier value of zero. An RTCP message may contain more than one BYE option. Multiple identifiers in a single BYE option are not allowed, to avoid ambiguities between the special value of zero and any necessary padding.

6.4 QOS: Quality of service measurement

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|F|  QOS = 36    |    length     |   reserved    |   reserved    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       packets expected                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       packets received                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    minimum delay (seconds)    |   minimum delay (fraction)    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    maximum delay (seconds)    |   maximum delay (fraction)    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    average delay (seconds)    |   average delay (fraction)    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| type=PORT (2) |    length     |  transport address           ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| type=ADDR (1) |    length     |   reserved    | address type  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     network-layer address                    ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The QOS option conveys statistics on the reception of packets from a single synchronization source on a single channel. These statistics are the number of packets received, the number of packets expected, the minimum delay, the maximum delay and the average delay. The expected number of packets may be computed according to the algorithm in Section A.5. The delay measures are in units of 1/65536 of a second, that is, with the same resolution as the timestamp in the fixed RTP header.

The synchronization source to which these statistics correspond is identified by appending to the fixed-length part of the QOS option the PORT and ADDR items, in that order, as received in an SDES option from that source. Together, the PORT and ADDR items form a globally unique identifier (as described with the SDES option, Section 6.2). If the source has supplied more than one ADDR item, only the first one from the SDES (the one immediately following the PORT item) is used. If no SDES option, or none with PORT and ADDR items, has been received from a particular source, the QOS option cannot be sent unless the PORT and ADDR items are known by some other mechanism.

The QOS option may be sent in either normal (forward) or reverse RTP packets. In the first case, the channel to which these statistics correspond is same as the channel on which the QOS option is sent; that is, the channel is identified by the destination (multicast or unicast) address, destination port and channel ID. If the QOS option is sent in a reverse RTP packet, the channel is identified by the channel ID in the header and the destination port number as the packet arrives at the synchronization source, which will be the same port that the source uses to send data on that channel, as described in Section 5.4. Sending the QOS option by multicast has the advantage that all participants in the session can compare their reception to that of others, and allows participants to adjust the rate at which QOS is sent based on the number of participants.

A single RTCP packet may contain several QOS options for different sources. It is left to the implementor to decide how often to transmit QOS options and which sources are to be included.

7 Security Considerations

RTP suffers from the same security liabilities as the underlying protocols. For example, an impostor can fake source or destination network addresses, or change the header or payload. For example, the SDES fields may be used to impersonate another participant. In addition, RTP may be sent via IP multicast, which provides no direct means for a sender to know all the receivers of the data sent and therefore no measure of privacy. Rightly or not, users may be more sensitive to privacy concerns with audio and video communication than they have been with more traditional forms of network communication [11]. Therefore, the use of security mechanisms with RTP is important.

As a first step, RTP options make it easy for all participants in a session to identify themselves; if deemed important for a particular application, it is the responsibility of the application writer to make listening without identification difficult. It should be noted, however, that privacy of the payload can generally be assured only by encryption.

The security options described in Section 5.5 can be used to implement message integrity, authentication and confidentiality and the combination of the three. These security services might also be provided at the IP layer as security mechanisms are developed for that layer.

The periodic transmission of SDES options from sources that are otherwise idle may make it possible to detect denial-of-service attacks, as the receiver can detect the absence of these expected messages. The messages that are received must be verified for integrity and authenticated before being accepted for this purpose.

Unlike for other data, ciphertext-only attacks may be more difficult for compressed audio and video sources. Such data is very close to white noise, making statistics-based ciphertext-only attacks difficult. Even without message integrity check options, it may be difficult for an attacker to detect automatically when he or she has found the secret cryptographic key since the bit pattern after correct decryption may not look significantly different from one decrypted with the wrong key. However, the session information is more or less constant and predictable, allowing known-plaintext attacks. Chosen-plaintext attacks appear, in general, to be difficult.

The integrity of the timestamp in the fixed RTP header can be protected by the message integrity options. If clocks are known to be synchronized, an attacker only has a very limited time window of maybe a few seconds every 18 hours to replay recorded RTP without detection by the receiver.

Key distribution and certificates are outside the scope of this document.

·8 RTP over Network and Transport Protocols

This section describes issues specific to carrying RTP packets within particular network and transport protocols.

8.1 Defaults

The following rules apply unless superseded by protocol-specific subsections in this section. The rules apply to both forward and reverse RTP packets.

RTP packets contain no length field or other delineation, therefore a framing mechanism is needed if they are carried in underlying protocols that provide the abstraction of a continuous bit stream rather than messages (packets). TCP is an example of such a protocol. Framing is also needed if the underlying protocol may contain padding so that the extent of the RTP payload cannot be determined. For these cases, each RTP packet is prefixed by a 32-bit framing field containing the length of the RTP packet measured in octets, not including the framing field itself. If an RTP packet traverses a path over a mixture of octet-stream and message-oriented protocols, each RTP-level bridge between these protocols is responsible for adding and removing the framing field.

A profile may specify that this framing method is to be used even when RTP is carried in protocols that do provide framing in order to allow carrying several RTP packets in one lower-layer protocol data unit, such as a UDP packet. Carrying several RTP packets in one network or transport packet reduces header overhead and may simplify synchronization between different streams.

8.2 ST-II

When used in conjunction with RTP, ST-II [12] service access ports (SAPs) have a length of 16 bits. The next protocol field (NextPCol, Section 4.2.2.10 in RFC 1190) is used to distinguish two encapsulations of RTP over ST-II. The first uses NextPCol value TBD and directly places the RTP packet into the ST-II data area. If NextPCol value TBD is used, the RTP header is preceded by a 32-bit header shown below. The octet count determines the number of octets in the RTP header and payload to be checksummed, allowing the checksum to cover only the header if it is preferred to ignore errors in the data. The 16-bit checksum uses the TCP and UDP checksum algorithm.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| count of octets to be checked |           checksum            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       RTP packet (fixed header, options and payload)         ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

9 RTP Profiles

RTP may be used for a variety of applications with somewhat differing requirements. The flexibility to adapt to those requirements is provided by allowing multiple choices in the main protocol specification, then defining a profile to select the appropriate choices for a particular class of applications and environment. A profile for audio and video applications may be found in the companion Internet draft draft-ietf-avt-profile.

Within this specification, the following possible uses of a profile have been identified, but this list is not meant to be exclusive:

- Define a set of formats (e.g., media encodings) and a default mapping of those formats to format values.
- efine new, application-class-specific options, or specify that an option, such as BOS, should be included in every packet.
- Specify the support for and semantics of particular options to be used in Reverse Path messages.
- Define new application-class-specific SDES items, or the data format, preferred use, or required use of

particular SDES items.
- Define when SDES applies to some grouping of channels rather than just a single channel.
- Specify that globally synchronized time is required for operation of an application.
- Specify that a particular underlying network or transport layer protocol will be used to carry RTP packets.
- Specify that the RTP header is always to be prefixed by the framing field to allow carrying multiple RTP packets (perhaps for different media) in one lower-layer packet.
- Describe the application of the quality-of-service option.

It is not expected that a new profile will be required for every application. Within one application class, it would be better to extend an existing profile rather than make a new one. For example, additional options or formats can be defined and registered through IANA for publication in the Assigned Numbers RFC as an alternative to publishing a new profile specification.

It is recommended that a profile specify a default port number to be used with that profile so that applications that support operation under multiple profiles can use the port number to select the profile.

A Implementation Notes

We describe aspects of the receiver implementation in this section. There may be other implementation methods that are faster in particular operating environments or have other advantages. These implementation notes are for informational purposes only.

The following definitions are used for all examples; the structure definitions are valid for 32-bit big-endian architectures only. Bit fields are assumed to be packed tightly, with no additional padding.

```
#include

typedef double CLOCK_t;

/* the definitions below are valid for 32-bit architectures and will
   have to be changed for 16- or 64-bit architectures */
typedef u_long  u_int32;
typedef u_short u_int16;

typedef enum {
  RTP_CSRC    = 0,
  RTP_SSRC    = 1,
  RTP_SDST    = 2,
  RTP_BOS     = 3,
  RTP_ENC     = 8,
  RTP_MIC     = 9,
  RTP_MICA    = 10,
  RTP_MICK    = 11,
  RTP_MICS    = 12,
  RTP_FMT     = 32,
  RTP_SDES    = 34,
  RTP_BYE     = 35,
  RTP_QOS     = 36,
  RTP_MINFMT = 96,
  RTP_MAXFMT = 126,
  RTP_APP     = 127
} rtp_option_t;

typedef struct {
  unsigned int ver:2;      /* version number */
  unsigned int channel:6;  /* channel id */
  unsigned int p:1;        /* option present */
  unsigned int s:1;         /* sync bit */
  unsigned int format:6;    /* format of payload */
  u_int16 seq;              /* sequence number */
  u_int32 ts;               /* timestamp */
} rtp_hdr_t;

typedef union {
  struct {                  /* generic first 16 bits of options */
    unsigned int final:1;   /* final option */
    unsigned int type:7;    /* option type */
  } generic;
  struct {
    unsigned int final:1;   /* final option */
    unsigned int type:7;    /* option type */
    u_char length;          /* length, including type/length */
    u_int16 id[1];          /* content source identifier */
  } csrc;
  /* ... */
} rtp_t;
```

A.1 Timestamp Recovery

For some applications it is useful to have the receiver reconstruct the sender's high-order bits of the NTP timestamp from the received 32-bit RTP timestamp. The following code uses double-precision floating point numbers for whole numbers with a 48-bit range. Other type definitions of CLOCK_t may be appropriate for different operating environments, e.g., 64-bit architectures or systems with slow floating point support. The routine applies to any clock frequency, not just the RTP value of 65,536 Hz, and any clock starting point. It will reconstruct the correct high-order bits as long as the local clock now is within one half of the wrap-around time of the 32-bit timestamp, e.g., approximately 9.1 hours for RTP timestamps.

```
#include

#define MOD32bit 4294967296.
#define MAX31bit 0x7fffffff

CLOCK_t clock_extend(ts, now)
u_int32 ts;   /* in: timestamp, low-order 32 bits */
CLOCK_t now;  /* in: current local time */
{
  u_int32 high, low;   /* high and low order bits of 48-bit clock */

  low  = fmod(now, MOD32bit);
  high = now / MOD32bit;

  if (low > ts) {
    if (low - ts > MAX31bit) high++;
  }
  else {
    if (ts - low > MAX31bit) high--;
  }
  return high * MOD32bit + ts;
} /* extend_timestamp */
```

Using the full timestamp internally has the advantage that the remainder of the receiver code does not have to be concerned with modulo arithmetic. The current local time does not have to be derived directly from the system clock for every packet. For audio samples, for example, it is more accurate to maintain the time within a synchronization unit in samples, incrementing by the number of samples per packet, and then converting to an RTP timestamp. The following code implements the conversion from a 8 KHz sample clock into an RTP timestamp. This assumes that the sample clock is also started at the RTP clock epoch (January 1, 1970). If not, the appropriate offset has to be added.

```
CLOCK_t t;        /* 8-kHz sample clock */
CLOCK_t RTP_ts;   /* RTP timestamp */

RTP_ts = floor(t * 8.192 + 0.5);
```

The whole seconds within NTP timestamps can be obtained by adding 2208988800 to the value of the standard Unix clock (generated, for example, by the gettimeofday system call), which starts from the year 1970. For the RTP timestamp, only the least significant 16 bits of the seconds are used.

A.2 Detecting the Beginning of a Synchronization Unit

RTP packets contain a bit flag indicating the end of a synchronization unit. The following code fragment determines, based on sequence numbers, if a packet is the beginning of a synchronization unit. It assumes that the packet header has been converted to host byte order.

```
static u_int32 seq_eos;
rtp_hdr_t *h;
static int flag;

if (h->s) {
  flag    = 1;
  seq_eos = h->seq;
}
/* handle wrap-around of sequence number */
else if (flag && (h->seq - seq_eos < 32768)) {
  flag = 0;
  /* code here to handle beginning of synchronization unit */
}
```

A.3 Demultiplexing and Locating the Synchronization Source

The combination of destination address, destination port and channel ID determines the channel. For each channel, the receiver maintains a list of all sources, content and synchronization sources alike, in a table or other suitable data structure. Synchronization sources are stored with a content source identifier of zero. When an RTP packet arrives, the receiver determines its network source address and port (from information returned by the operating system), synchronization source identifier (SSRC option) and content source identifier(s) (CSRC option). To locate the table entry containing timing information, mapping from content descriptor to actual encoding, etc., the receiver sets the content source identifier to zero and locates a table entry based on the tuple (transport source address, synchronization source identifier, 0).

The receiver identifies the contributors to the packet (for example, the speaker who is heard in the packet) through the list of content source identifiers carried in the CSRC option. To locate the table entry, it matches on the triple (network address and port, synchronization source identifier, content source identifier).

A.4 Parsing RTP Options

The following code segment walks through the RTP options, preventing infinite loops due to zero and invalid length fields. Structure definitions are valid for big-endian architectures only.

```
u_int32 len;       /* length of RTP packet in bytes */
u_int32 *pt;       /* pointer */
rtp_hdr_t *h;      /* fixed header */
rtp_t *r;          /* options */

if (h->p) {
  pt = (u_int32 *)(h+1);
  do {
    r = (rtp_t *)pt;
    pt += r->generic.length;   /* point to end of option */

    /* invalid length field */
    if ((char *)pt - (char *)h > len || r->generic.length == 0) return -1;

    switch(r->generic.type) {
      case RTP_BYE:
        /* handle BYE option */
        break;
      case RTP_CSRC:
        /* handle CSRC option */
        break;

        /* ... */

      default:
                       if    (r->generic.type    >=    RTP_MINFMT    &&    r-
>generic.type <= RTP_MAXFMT) {
          /* call option handler particular to this format */
          (parse_format_options[h->format])(r);
        }
        else break;     /* ignore undefined options */
    }
  } while (!r->generic.final);
}
```

A.5 Determining the Expected Number of RTP Packets

The number of packets expected can be computed by the receiver by tracking the first sequence number received (seq0), the last sequence number received, seq, and the number of complete sequence number cycles:

```
expected = cycles * 65536 + seq - seq0 + 1;
```

The cycle count is updated for each packet, where seq_prior is the sequence number of the prior packet:

```
unsigned long seq, seq_prior;

if (seq - seq_prior > 65536)
  cycle++;
else if (seq - seq_prior > 32768)
  cycle--;

seq_prior = seq;
```

Acknowledgments

This memorandum is based on discussions within the IETF Audio/Video Transport working group chaired by Stephen Casner. The current protocol has its origins in the Network Voice Protocol and the Packet Video Protocol (Danny Cohen and Randy Cole) and the protocol implemented by the vat application (Van Jacobson and Steve McCanne). Stuart Stubblebine (ISI) helped with the security aspects of RTP. Ron Frederick (Xerox PARC) provided extensive editorial assistance.

B Addresses of Authors

Stephen Casner
USC/Information Sciences Institute

4676 Admiralty Way
Marina del Rey, CA 90292-6695
telephone: +1 310 822 1511 (extension 153)
electronic mail: casner@isi.edu

Henning Schulzrinne
AT&T Bell Laboratories
MH 2A244
600 Mountain Avenue
Murray Hill, NJ 07974-0636
telephone: +1 908 582 2262
facsimile: +1 908 582 5809
electronic mail: hgs@research.att.com

References

[1] D. E. Comer, Internetworking with TCP/IP, vol. 1. Englewood Cliffs, New Jersey: Prentice Hall, 1991.

[2] J. Postel, "Internet protocol," Network Working Group Request for Comments RFC 791, Information Sciences Institute, Sept. 1981.

[3] International Standards Organization, "ISO/IEC DIS 10646-1:1993 information technology -- universal multiple-octet coded character set (UCS) -- part I: Architecture and basic multilingual plane," 1993.

[4] The Unicode Consortium, The Unicode Standard. New York, New York: Addison-Wesley, 1991.

[5] D. L. Mills, "Network time protocol (version 3) -- specification, implementation and analysis," Network Working Group Request for Comments RFC 1305, University of Delaware, Mar. 1992.

[6] S. Kent, "Understanding the Internet certification system," in Proceedings of the International Networking Conference (INET), (San Francisco, California), pp. BAB--1 -- BAB--10, Internet Society, Aug. 1993.

[7] D. Balenson, "Privacy enhancement for internet electronic mail: Part III: Algorithms, modes, and identifiers," Network Working Group Request for Comments RFC 1423, IETF, Feb. 1993.

[8] V. L. Voydock and S. T. Kent, "Security mechanisms in high-level network protocols," ACM Computing Surveys, vol. 15, pp. 135--171, June 1983.

[9] J. Kaliski, Burton S., "The MD2 message-digest algorithm," Network Working Group Request for Comments RFC 1319, RSA Laboratories, Apr. 1992.

[10] R. Rivest, "The MD5 message-digest algorithm," Network Working Group Request for Comments RFC 1321, IETF, Apr. 1992.

[11] S. Stubblebine, "Security services for multimedia conferencing," in 16th National Computer Security Conference, (Baltimore, MD), pp. 391--395, Sept. 1993.

[12] C. Topolcic, S. Casner, C. Lynn, Jr., P. Park, and K. Schroder, "Experimental internet stream protocol, version 2 (ST-II)," Network Working Group Request for Comments RFC 1190, BBN Systems and Technologies, Oct. 1990.